```
0630 --    1  (*****************************************************************************)
0630 --    2  (*                                                                           *)
0630 --    3  (*                    PASCAL TO LAMBDA-CALCULUS  COMPILER:                   *)
0630 --    4  (*                    -----------------------------------                    *)
0630 --    5  (*                                                                           *)
0630 --    6  (*    AUTHOR:  ERICH KALTOFEN                                 DATE: MAY 1979 *)
0630 --    7  (*    ADDRESS: RENSSELEAR POLYTECHNIC INSTITUTE                              *)
0630 --    8  (*             DEPT. OF MATHEMATICAL SCIENCES                                *)
0630 --    9  (*             TROY, NEW YORK 12181                                          *)
0630 --   10  (*    VERSION: 1.A                                                           *)
0630 --   11  (*    THIS WORK WAS PARTIALLY SUPPORTED BY U.S. OFFICE OF NAVAL RESEARCH     *)
0630 --   12  (*    CONTRACT NO. N00014-75-C-1026.                                         *)
0630 --   13  (*    TO OBTAIN FURTHER DOCUMENTATION FOR THIS PROGRAM WRITE TO THE AD-      *)
0630 --   14  (*    DRESS GIVEN ABOVE.                                                     *)
0630 --   15  (*                                                                           *)
0630 --   16  (*****************************************************************************)
0630 --   17
0630 --   18
0630 --   19  (*$S- DO NOT FLAG GLOBALLY DECLARED CONTROL VARIABLES *)
0630 --   20  (*$X+ LIST CROSS-REFERENCE *)
0630 --   21  PROGRAM LAMBDA1A(
0630 --   22
0630 --   23
0630 --   24  (* N O T I C E:                                                            *)
0630 --   25  (* THE FOLLOWING SOURCE INCLUDES TRACE COMMANDS OF DIFFERRENT LEVELS,      *)
0630 --   26  (* EACH OF WHICH IS DISABLED BY EMBRACING IT IN A TYPICAL WAY (COMMENT     *)
0630 --   27  (* PARENTHESIS FOLLOWED BY 'DE-I' OR PRECEEDED BY 'I-BUG' RESP. WHERE     *)
0630 --   28  (* I IS A LEVEL BETWEEN 0 AND 4). ONE SHOULD USE A TEXT EDITOR TO ALTER   *)
0630 --   29  (* THESE BRACES AND ENABLE DEBUGGING COMMANDS ON WANTED LEVELS TO BE       *)
0630 --   30  (* COMPILED. THE TRACE INFORMATION IS WRITTEN ON THE EXTERNAL FILE         *)
0630 --   31  (* 'DEBUG' AND THE LEVEL NUMBERS WILL PERFORM THE FOLLOWING:              *)
0630 --   32  (*    0...MUST ALWAYS BE ENABLED IF ANY TRACING IS DONE. MAY PRODUCE      *)
0630 --   33  (*        CRUCIAL INFORMATION IN CASE OF COMPILER FAILURE                  *)
0630 --   34  (*    1...TRACES LEXICAL SCANNER. WILL PRODUCE A LOT OF OUTPUT            *)
0630 --   35  (*    2...TRACES PARSER. IMPORTANT FOR LOCATING AN ERROR                  *)
0630 --   36  (*    3...TRACES TYPE CHECKING AND OTHER SEMANTIC PROCESSES               *)
0630 --   37  (*    4...TRACES CODE GENERATOR                                           *)
0630 --   38  (* MANY OF THE TRACE COMMANDS WILL NOT BE LISTED  LATERON                 *)
0630 --   39
0630 --   40                    (* EXTERNAL FILES *)
0630 --   41                    INPUT,  (* PASCAL SOURCE *)
0644 --   42                    OUTPUT, (* LISTING, CROSS REFERENCE, ERROR MESSAGES *)
0644 --   43                    SPUNCH (* LAMBDA CODE OF PROGRAM *)
0644 --   44                    (*DE-0 ,DEBUG 0-BUG*)
0644 --   45                   );
0644 --   46
0644 --   47  LABEL 9999; (* EPILOG OF COMPILATION *)
0644 --   48
0644 --   49  CONST (* C O N S T A N T S *)
0644 --   50        (* GLOBAL SIZE PARAMETERS *)
0644 --   51        (* MAKE CHANGES HERE ONLY IF NOT ENOUGH SPACE *)
0644 --   52        MAXNRID=500; (* UPPER BOUND OF NUMBER OF IDENTIFIERS *)
0644 --   53        MAXSTRGL=500; (* MAXMUM TOTAL LENGTHS OF LITERALS *)
0644 --   54        MAXWSL=10000; (* WORKSPACE FOR STRING ROUTINES (CODE GENERATION) *)
0644 --   55        MAXLNL=100; (* MAXIMUM LENGTH OF INPUT LINE *)
0644 --   56        MAXCODLNL=72; (* MAXIMUM LENGTH OF CODE LINES ON SPUNCH *)
0644 --   57        MAXPAGEL=60; (* MAXIMUM NUMBER OF LINES PER PAGE *)
0644 --   58        NRBUCK=253; (* NUMBER OF BUCKETS FOR HASHING *)
```

```
0644 --   59            MAXNRERR=50; (* MAXIMUM NUMBER OF ERROR MESSAGES *)
0644 --   60            MAXCOMPNST=3; (* MAXIMUM DESCENT FOR POINTER TYPE COMPARISIONS *)
0644 --   61
0644 --   62            (* SOME SPECIAL CONSTANTS *)
0644 --   63            NRKEYW=35; (* NUMBER OF KEYWORDS *)
0644 --   64            LOWORD=0; (* LOWEST ORDER OF CHARACTERS *)
0644 --   65            HIGORD=255; (* HIGHEST ORDER OF CHARACTERS *)
0644 --   66            ORDEOLNCH=37; (* ORDER OF EOLN-CHARACTER *)
0644 --   67            ORDEOSTRGCH=3; (* ORDER OF STRING-TERMINATOR *)
0644 --   68            ORDDBLDOTCH=255; (* ORDER OF DOUBLEDOT-CHARACTER *)
0644 --   69
0644 --   70            (* MNEMONICS FOR TOKENNUMBERS *)
0644 --   71            ANDSYM    = 1; ARRAYSYM = 2; BEGINSYM = 3; CASESYM  = 4; CONSTSYM = 5;
0644 --   72            DIVSYM    = 6; DOSYM    = 7; DOWNTOSYM= 8; ELSESYM  = 9; ENDSYM   =10;
0644 --   73            FILESYM   =11; FORSYM   =12; FUNCSYM  =13; GOTOSYM  =14; IFSYM    =15;
0644 --   74            INSYM     =16; LABELSYM =17; MODSYM   =18; NILSYM   =19; NOTSYM   =20;
0644 --   75            OFSYM     =21; ORSYM    =22; PACKEDSYM=23; PROCSYM  =24; PROGSYM  =25;
0644 --   76            RECORDSYM=26; REPEATSYM=27; SETSYM   =28; THENSYM  =29; TOSYM    =30;
0644 --   77            TYPESYM   =31; UNTILSYM =32; VARSYM   =33; WHILESYM =34; WITHSYM  =35;
0644 --   78            IDENTIFIER=40;
0644 --   79            LPARASYM =46; RPARASYM =47; LBRACKSYM=48; RBRACKSYM=49; SEMICSYM =50;
0644 --   80            COMMASYM =51; POINTER  =52; EQUALSYM =53; PERIODSYM=54; DOUBLEDOT=55;
0644 --   81            COLONSYM =56; BECOMES  =57; STRINGSYM=58; UNSGINTEG=59; UNSGREAL =60;
0644 --   82            PLUSMINUS=61; MULTOPER =62; RELOPER  =63;
0644 --   83
0644 --   84            (* CLASSIFICATIONS OF INPUT CHARACTERS *)
0644 --   85            DIGIT     = 1; PERIODCHR= 2; ELETTER  = 3; PLUS      = 4; MINUS     = 5;
0644 --   86            STAR      = 6; SLASH    = 7; LPARACHR = 8; RPARACHR = 9; LBRACKCHR=10;
0644 --   87            RBRACKCHR=11; SEMICCHR =12; COMMACHR =13; AMPERSAND=14; EQUALCHR =15;
0644 --   88            LESSCHR   =16; GREATRCHR=17; COLONCHR =18; QUOTE     =19; BLANK     =20;
0644 --   89            EOLNCHR   =21; LBRACECHR=22; LETTER   =23; DBLDOTCHR=24; OTHERCHR =25;
0644 --   90
0644 --   91       TYPE (* G L O B A L  D A T A - S T R U C T U R E S *)
0644 --   92
0644 --   93
0644 --   94       (* 1. LEXICAL SCANNER *)
0644 --   95
0644 --   96            HASHPTR=0..MAXNRID; (* RANGE OF HASH TABLE *)
0644 --   97            TOKENRG=0..RELOPER;
0644 --   98
0644 --   99       (* 2. PARSER AND SEMANTIC ANALYSER *)
0644 --  100
0644 --  101            IDCLASS=(SCALID,TYPID,VARID,FLDID,PROCID,FUNCID);
0644 --  102            TYPCLASS=(SCALTYP,ARRTYP,RECTYP,PTRTYP,UNDEF);
0644 --  103
0644 --  104            (* RECORDS FOR BOOKKEEPING PURPOSES *)
0644 --  105            ACTIVS=RECORD (* ACTIVE IDENTIFIER DESCRIPTION *)
0644 --  106                      IDNR: HASHPTR; (* IDENTIFIER NAME REF. *)
0644 --  107                      LEVNR: INTEGER; (* LEVEL OF OCCURENCE *)
0644 --  108                      PARM: (* KIND OF PARAMETER (VAR-, PROC-, FUNCIDS) *)
0644 --  109                            (VARB, (* VARIABLE *)
0644 --  110                             VALP, (* VALUE PARAMETER *)
0644 --  111                             REFP); (* REFERENCE PARAMETER *)
0644 --  112                      XREF: @XREFS; (* CROSS REFERENCE LIST *)
0644 --  113                      NXTACT: @ACTIVS; (* LINK *)
0644 --  114                      CASE IDKLASS: IDCLASS OF
0644 --  115                      SCALID: (STYP: @TYPES;
0644 --  116                               VALUU: INTEGER (* CARDINALITY OF SCALAR *));
```

```
0644 --  117                    TYPID,FLDID,VARID: (TYP: @TYPES);
0644 --  118                    PROCID, FUNCID: (ARG: @ARGS; (* ARGUMENT TYPE LIST *)
0644 --  119                                     RETTYP: @TYPES; (* RETURN TYPE *)
0644 --  120                                     SWFORW: BOOLEAN; (* FORWARD DECLARED? *)
0644 --  121                                     FORWARGS: @ACTIVS (* FORWARD ARGUMENTS *))
0644 --  122                 END;
0644 --  123
0644 --  124           TYPES=RECORD (* TYPEDESCRIPTION *)
0644 --  125                   CASE TYPKLASS: TYPCLASS OF
0644 --  126                   SCALTYP: (SCIDNR: HASHPTR; (* TYPE ID OF IT *)
0644 --  127                             SCLEVNR: INTEGER; (* LEVEL OF DECLARATION *)
0644 --  128                             LOWBND, HIGBND: INTEGER (* RANGE *));
0644 --  129                   ARRTYP: (INDXTYP: @TYPES; (* INDEX TYPE *)
0644 --  130                            COMPTYP: @TYPES; (* COMPONENT TYPE *)
0644 --  131                            SWPACKA: BOOLEAN (* PACKED OR NOT *));
0644 --  132                   RECTYP: (SECTNS: @ACTIVS; (* LIST OF FIELDS *)
0644 --  133                            SWPACKR: BOOLEAN; (* WAS IT PACKED OR NOT *)
0644 --  134                            SWXRFPRR: BOOLEAN (*FLAG FOR PRINTING FIELD IN X-REF*) );
0644 --  135                                              (*TRUE IF IMPLICIT DEFINITION (VAR) *)
0644 --  136                   PTRTYP: (PTRIDNR: HASHPTR; (* REF. TYPE ID *)
0644 --  137                            REFERTYP: @TYPES (* REF. TYPE *));
0644 --  138                   UNDEF: (UNDFIDNR: HASHPTR (* ITS NAME, IF WE KNOW *) );
0644 --  139                END;
0644 --  140
0644 --  141           (* LIST OF EXPRESSION CODES AND TYPES           *)
0644 --  142           (* USED FOR ACTUAL PARAMETERS AND ARRAY INDICES *)
0644 --  143           EXPLST=RECORD
0644 --  144                    EXPLSTCOD: @STRGDESC;(*CHARVAR*)  (* CODE FOR EXPRESSION *)
0644 --  145                    EXPLSTTYP: @TYPES; (* ITS IMLICIT TYPE *)
0644 --  146                    SWVAR: BOOLEAN; (* SWITCH IF IT IS A VARIABLE *)
0644 --  147                    NXTEXP: @EXPLST (* LINK POINTER *)
0644 --  148                END;
0644 --  149
0644 --  150           (* LIST OF X-REFS *)
0644 --  151           XREFS=RECORD OCC: INTEGER; (* WHERE OCCURED *)
0644 --  152                        NXTREF: @XREFS (* LINK *) END;
0644 --  153
0644 --  154           (* LIST OF IDENTIFIERS *)
0644 --  155           IDLST=RECORD ID: HASHPTR; NXTID: @IDLST (* LINK *) END;
0644 --  156
0644 --  157           ARGS=RECORD (* FORMAL PARAMETER DESCRIPTION *)
0644 --  158                   ARGTYP: @TYPES; (* TYPE OF PARAMETER *)
0644 --  159                   PASSKND: BOOLEAN; (* VAR PARAMETER OR NOT *)
0644 --  160                   NXTARG: @ARGS (* LINK *)
0644 --  161                END;
0644 --  162
0644 --  163           (* LIST OF IDENTIFIER RECORDS *)
0644 --  164           ACTLST=RECORD ACTLSTPTR: @ACTIVS; NXTONE: @ACTLST (* LINK *) END;
0644 --  165
0644 --  166           (* LIST OF VALUES OF A CASE LABEL LIST *)
0644 --  167           VALUS=RECORD VALEE: INTEGER; NXTVAL: @VALUS (* LINK *) END;
0644 --  168
0644 --  169           (* LIST OF  A L L  POINTER REFERENCES *)
0644 --  170           FORWLST=RECORD LOCTYP: @TYPES; (* WHERE POINTER TYPE LOC. *)
0644 --  171                          LOCSCNR: INTEGER; (* WHAT TYPE IT WAS *)
0644 --  172                          NEXTF: @FORWLST  (* LINK *) END;
0644 --  173
0644 --  174      (* 3. CODE GENERATOR *)
```

```
0644 --       175
0644 --       176          (* DOUBLE-LINKED LIST FOR (RE-USABLE) STACK OF BOOLEANS *)
0644 --       177          (* IT IS USED AS GLOBAL STACK TO DETERMINE IF EXPRESSIONS ARE VARIABLES *)
0644 --       178          DBLBOOL=RECORD SWVAR: BOOLEAN; (* TRUE IF EXPRESSION SINGLE VARIABLE *)
0644 --       179                         UP, DOWN: @DBLBOOL (* DOUBLE LINKS *) END;
0644 --       180
0644 --       181          (* DESCRIPTOR FOR STRINGS *)
0644 --       182          STRGDESC=RECORD
0644 --       183                      WSPOS: 1..MAXWSL; (* STARTING POINT OF STRING *)
0644 --       184                      MAXLG: INTEGER; (* LENGTH RESERVED FOR STRING *)
0644 --       185                                      (* NEGATIVE IF STRING UNUSED  *)
0644 --       186                      CURLG: INTEGER; (* CURRENT LENGTH *)
0644 --       187                      NXTSTRG: @STRGDESC (* LINK POINTER *)
0644 --       188                    END;
0644 --       189          CHARVAR=@STRGDESC; (* SUGGESTIVE ALIAS *)
0644 --       190
0644 --       191          (* LIST OF SUBSCRIPTS AND BOUNDS *)
0644 --       192          SUBSBNDS=RECORD (* USED FOR ASSIGNMENT TO ARRAY ELEMENT *)
0644 --       193                      SUB: CHARVAR; (* LE FOR SUBSCRIPT *)
0644 --       194                      BND: CHARVAR; (* LE FOR UPPER BOUND (ALREADY SHIFTED) *)
0644 --       195                      NXTSBND: @SUBSBNDS; (* LINK POINTER *)
0644 --       196                    END;
0644 --       197
0644 --       198       VAR (* G L O B A L   V A R I A B L E S *)
0644 --       199
0644 --       200
0644 --       201       (* 1. LEXICAL SCANNER AND ERROR GENERATOR *)
0644 --       202
0644 --       203            LINE: PACKED ARRAY(.1..MAXLNL.) OF CHAR; (* INPUT LINE BUFFER *)
06A8 --       204            LNPOS: 0..MAXLNL; (* CURRENT POSITION IN LINE *)
06AC --       205            SWEOLN: BOOLEAN; (* TRUE IF CURRENTLY AT CARRIAGE RETURN *)
06B0 --       206
06B0 --       207            SCNR: INTEGER; (* SEMICOLON COUNTER *)
06B4 --       208            PAGENR: INTEGER; (* PAGE COUNTER *)
06B8 --       209            LNCNT: 1..MAXPAGEL; (* LINE NUMBER ON CURRENT PAGE *)
06BC --       210            HEADER: PACKED ARRAY(.1..45.) OF CHAR; (* HEADER TEXT *)
06E9 --       211            PROGID: HASHPTR; (* PROGAM NAME *)
06F0 --       212
06F0 --       213            KEYW: ARRAY(.1..NRKEYW.) OF PACKED ARRAY(.1..10.) OF CHAR; (* KEYWORDS *)
084E --       214            (* NAMES FOR TOKENS OTHER THAN KEYWORDS *)
084E --       215            TOKW: ARRAY(.IDENTIFIER..RELOPER.) OF PACKED ARRAY(.1..10.) OF CHAR;
0940 --       216            IDREG: PACKED ARRAY(.1..10.) OF CHAR; (* TRUNCATED INPUT-IDENTIFIER *)
094A --       217            INSERTWORD: PACKED ARRAY(.1..10.) OF CHAR; (* FOR ERROR MESSAGES *)
0956 --       218
0956 --       219            CHRSEL: ARRAY(.CHAR.) OF DIGIT..OTHERCHR; (* CASE FOR INPUT CHARACTER *)
0D58 --       220            (* TRANSITION TABLE FOR NUMBER SCANNER *)
0D58 --       221            TRANSIT: ARRAY(.1..6,1..6,1..2.) OF INTEGER;
0E78 --       222            STRING: PACKED  ARRAY(.1..MAXSTRGL.) OF CHAR; (* STORAGE FOR LITERALS *)
106C --       223            STRGPTR: 1..MAXSTRGL; (* CURRENT POSITION IN LITERAL STORAGE *)
1070 --       224
1070 --       225            (* HASH TABLE ENTRIES (TRUNCATED IDS) *)
1070 --       226            HASHNAME: ARRAY(.HASHPTR.) OF PACKED ARRAY(.1..10.) OF CHAR;
2402 --       227            HASHNEXT: ARRAY(.HASHPTR.) OF HASHPTR; (* LINK POINTERS OF HASH TABLE *)
2BD8 --       228            FREESPACE: HASHPTR; (* INDEX OF LEFT FREE ELEMENTS IN HASH TABLE *)
2BDC --       229
2BDC --       230            ERRORS: FILE OF CHAR; (* INTERNAL FILE FOR ACCUMULATING ERROR MESSAGES *)
2BF0 --       231            (* ERROR TEXT (IF PUZZLED TOGETHER) *)
2BF0 --       232            ERRDESC: PACKED ARRAY(.1..40.) OF CHAR;
```

```
2C18 --       233          ERRCNT: INTEGER; (* HOW MANY ERRORS SO FAR *)
2C1C --       234          SWERRMSG: BOOLEAN; (* FALSE IF NO ERROR SO FAR *)
2C20 --       235          (*DE-0 DEBUG: TEXT; 0-BUG*)
2C20 --       236
2C20 --       237          CH: CHAR; (* HOLDS CHARACTER READ BY NEXTCH *)
2C24 --       238          TOKENNR: TOKENRG; (* WHICH TOKEN (SEE MNEMONICS ABOVE) *)
2C28 --       239          TOKENFD: INTEGER; (* ASSOCIATED TOKEN VALUE AS FOLLOWS: *)
2C2C --       240                            (* IDENTIFIER:  HASHINDEX *)
2C2C --       241                            (* PLUSMINUS:   "+"(1), "-"(2) *)
2C2C --       242                            (* MULTOPER:    "*"(1), "/"(2), "DIV"(3), "MOD"(4), *)
2C2C --       243                            (*              "AND"(5) *)
2C2C --       244                            (* RELOPER:     "<>"(2), "<"(3), ">"(4), "<="(5), *)
2C2C --       245                            (*              ">="(6), "IN"(7) *)
2C2C --       246                            (* UNSGINTEG:   ITS ACTUAL VALUE *)
2C2C --       247                            (* UNSGREAL:    A TRUNCATED VALUE *)
2C2C --       248                            (* STRINGSYM:   A CODED FORM OF LENGTH AND STRING- *)
2C2C --       249                            (*              INDEX (SEE CASE QUOTE IN ADVANCE) *)
2C2C --       250
2C2C --       251      (* 2. PARSER AND SEMANTIC CHECKER *)
2C2C --       252
2C2C --       253          SYNCHRSYM: SET OF TOKENRG; (* SYNCHRONIZATION SYMBOLS FOR ERROR RECOV. *)
2C34 --       254          DONTSKIPSYM: SET OF TOKENRG; (* SYMBOLS NOT TO BE PASSED WHILE SYNCHRON. *)
2C3C --       255          NOTSKIP: BOOLEAN; (* FALSE IFF PARSER IN ERRORMODE *)
2C40 --       256
2C40 --       257          (* STACKS OF ACT.ID.-REC.-POINTERS ON DIFFERENT LEVELS FOR EACH ID *)
2C40 --       258          SRCHTAB: ARRAY(.HASHPTR.) OF @ACTLST;
3414 --       259
3414 --       260          LEVEL: INTEGER; (* GLOBAL LEVEL COUNTER *)
3418 --       261          STMNR: INTEGER; (* GLOBAL STATEMENT COUNTER *)
341C --       262          NEST: INTEGER; (* GLOBAL NESTING COUNT *)
3420 --       263
3420 --       264          STACK: @ACTIVS; (* STACK OF ALL ACT.ID.-RECORDS *)
3424 --       265
3424 --       266          ATTR: @ACTIVS; (* SORTED LIST OF DROPPED ACT.ID.-RECORDS AND REC.-FIELDS *)
3428 --       267          XRFF: FILE OF CHAR; (* INTERNAL FILE FOR BUILDING CROSS-REFERENCE *)
343C --       268                              (* SHOULD BE RELACED BY A CHARVAR STRING, WHICH *)
343C --       269                              (* WAS NOT DEVELOPED WHEN THIS PART WAS WRITTTEN *)
343C --       270
343C --       271          DUMMYFD: INTEGER; (* UNUSED RETURN ARGUMENT FOR SOME PARSING PROC. CALLS *)
3440 --       272          DUMMYTYP: @TYPES; (*                -- SAME AS ABOVE --                  *)
3444 --       273
3444 --       274          (* STANDARD IDENTIFIERS AND SOME NAMES *)
3444 --       275          SPINTEGER, SPREAL, SPCHAR, SPBOOLEAN, SPTRUE, SPFALSE, SPINPUT,
3444 --       276             SPOUTPUT, SPNEW, SPGET, SPPUT, SPORD, SPCHR, SPSUCC, SPPRED,
3444 --       277             SPREAD, SPREADLN, SPWRITE, SPWRITELN,
3444 --       278             SPLABEL, SPFILE, SPSET, SPFORWARD, SPCONST: HASHPTR;
34A4 --       279
34A4 --       280          (* SOME STANDARD TYPES *)
34A4 --       281          TYPINTEGER, TYPBOOLEAN, TYPCHAR: @TYPES;
34B0 --       282          NILTYP: @TYPES; (* TYPE FOR NIL-POINTER *)
34B4 --       283          UNDFZERO: @TYPES; (* NOT-QUALIFIED UNDEFINED TYPE (UNDFIDNR=0) *)
34B8 --       284
34B8 --       285          IMPLTYPNR: INTEGER; (* COUNTER FOR IMPLICIT TYPE NAMES *)
34BC --       286          FORWTYPS: @FORWLST; (* HEAD OF FORWARD-REFERENCE CHAIN *)
34C0 --       287
34C0 --       288          COMPNEST: 0..MAXCOMPNST; (* GLOBAL COUNT OF NESTINGS FOR COMPARE  *)
34C4 --       289          TOPBOOL: @DBLBOOL; (* STACK POINTER FOR VARIABLE-SWITCHES *)
34C8 --       290          ALLOWFW: BOOLEAN; (* SWITCH TO ALLOW FORWARD POINTER REFERENCES *)
```

```
34CC --       291
34CC --       292          (* FLAGS FOR COMPILER OPTIONS "S" AND "X" *)
34CC --       293          NOTSTAND: BOOLEAN; (* IF TRUE THEN ALLOW SOME NON-STANDARD SYNTAX *)
34D0 --       294          XREFLIST: BOOLEAN; (* IF TRUE PRINT LIST OF X-REFERENCE ON OUTPUT *)
34D4 --       295
34D4 --       296      (* 3. CODE GENERATOR *)
34D4 --       297
34D4 --       298          (* GLOBAL VARIABLES FOR STRING-MANIPULATING ROUTINES *)
34D4 --       299          WORKSPACE: PACKED ARRAY(.1..MAXWSL.) OF CHAR;
5BE4 --       300                     (* CONTAINS ALL CODE-STRINGS LITERALLY *)
5BE4 --       301          ALLOCS: @STRGDESC; (* LIST OF CURRENTLY ALLOCATED STRINGS *)
5BE8 --       302          EXTRAS: @STRGDESC; (* LEFTOVERS FROM COMPACTIFICATION *)
5BEC --       303          REMWS: INTEGER; (* POINTER TO REMAINING FREE WORKSPACE *)
5BF0 --       304
5BF0 --       305          SPUNCH: FILE OF CHAR; (* COMPILED CODE *)
5C04 --       306          DUMMYCOD: CHARVAR; (* SAME AS DUMMYFD, FOR CODE GENERATION *)
5C08 --       307          DUMMYSBND: @SUBSBNDS; (* DUMMY SECOND PARAMETER FOR VARSELECTS *)
5C0C --       308          (*DE-4 DEBUGJ: INTEGER; 4-BUG*)
5C0C --       309
5C0C --       310          (* GLOBAL LITERALS FOR CODE *)
5C0C --       311          RETRIEVE, REPLACE: CHARVAR; (* NAMES FOR THE STORAGE FUNCTIONS *)
5C14 --       312          DOLLAR, LITCOMMA, LITCOLON, LITBLANK: CHARVAR; (* "$", ",", ":", ' ' *)
5C24 --       313          PI: CHARVAR; (* "$PI$" RETURN VARIABLE PREFIX *)
5C28 --       314          STM: CHARVAR; (* "$STM" STATEMENT DEFINITION PREFIX *)
5C2C --       315          OMEGA: CHARVAR; (* "$OMEGA" UNDEFINED VALUE LE *)
5C30 --       316          EQUIV: CHARVAR; (* EQUIVALENT CHARACTER FOR LE DEFINITIONS *)
5C34 --       317          TERMIN: CHARVAR; (* TERMINATION CHARACTER FOR LE DEFINITIONS *)
5C38 --       318          VALPREF: CHARVAR; (* "$VAL$" PREFIX FOR VALUE PARAMETER ASSIGNMENTS *)
5C3C --       319          PROGSTMNR: INTEGER; (* STMNR OF OUTERMOST COMPOUND *)
5C40 --       320          SUPERSCR: BOOLEAN; (* COMPILER OPTION FLAG FOR "U" *)
5C44 --       321                    (* IF TRUE THEN ATTACHED SUPERSCRIBED NUMBERS TO PAIRS OF *)
5C44 --       322                    (* PARENTHESIS INSIDE GENERATED CODE FOR READABILITY SAKE *)
5C44 --       323
5C44 --       324      (* 5. MISCALENEOUS *)
5C44 --       325          I: INTEGER; (* GLOBAL INDEX FOR LOOPS WITHOUT SIDE-EFFECTS *)
5C48 --       326          DIGCHR: PACKED ARRAY(.0..9.) OF CHAR; (* CHARACTERS FOR DIGITS *)
5C52 --       327
5C52 --       328
5C52 --       329      (**************************************)
5C52 --       330      (* SOME GENERALLY UTILIZED PRODEDURES *)
5C52 --       331      (**************************************)
5C52 --       332
5C52 -- A     333      FUNCTION CONV(N: INTEGER; VAR DIGSTRG: PACKED ARRAY(.0..10.) OF CHAR): INTEGER;
0050 --       334      (* PUTS N IN CHARACTER-FORMAT LEFT-ALLIGNED INTO DIGSTRG *)
0050 --       335      (* DIGSTRG(.0.) CONTAINS ' ' IF N>0, '-' OTHERWISE       *)
0050 --       336      VAR ZEROCNT, I, J: 0..10; M: INTEGER; SWPLUS: BOOLEAN;
0000 0- A     337      BEGIN M:=ABS(N); SWPLUS:=(M=N);
003C --       338            IF SWPLUS THEN DIGSTRG(.0.):=' ' ELSE DIGSTRG(.0.):='-';
0062 --       339            FOR I:=10 DOWNTO 1 DO
0068 1-       340                BEGIN DIGSTRG(.I.):=DIGCHR(.M MOD 10.);
00D0 --       341                      M:=M DIV 10
00D0 -1       342                END;
00F2 --       343            ZEROCNT:=0; J:=1;
0100 --       344            WHILE (J<10) AND (DIGSTRG(.J.)='0') DO
014A 11       345               BEGIN ZEROCNT:=ZEROCNT+1; J:=J+1 END;
018A --       346            FOR I:=1 TO 10-ZEROCNT DO
019C --       347                DIGSTRG(.I.):=DIGSTRG(.ZEROCNT+I.);
0212 --       348            J:=1;
```

```
021A --       349            WHILE (J<=ZEROCNT) DO
0226 11       350               BEGIN DIGSTRG(.11-J.):=' '; J:=J+1 END;
026E --       351            CONV:=10-ZEROCNT
026E -0 A     352      END;
029C --       353
029C -- A     354      PROCEDURE NEWPAGE;
0040 --       355        VAR PROGNAME: PACKED ARRAY(.1..10.) OF CHAR;
0000 0- A     356        BEGIN PAGENR:=PAGENR+1;
0016 --       357              IF PROGID<>0 THEN PROGNAME:=HASHNAME(.PROGID.)
003E --       358                           ELSE PROGNAME:=' PASCAL K ';
0054 --       359              WRITELN(OUTPUT,'1',HEADER,' ***',PROGNAME,'***',
00A8 --       360                              'PAGE':MAXLNL-43,PAGENR:4);
00D0 --       361              WRITELN(OUTPUT,' '); LNCNT:=2
00E8 -0 A     362        END;
0128 --       363
0128 -- A     364      PROCEDURE LINECNT;
0000 0- A     365         BEGIN LNCNT:=LNCNT+1;
0028 --       366               IF LNCNT=MAXPAGEL THEN NEWPAGE
0034 -0 A     367         END;
0044 --       368
0044 -- A     369      PROCEDURE ERRMSG(CODE: INTEGER; TEXT: PACKED ARRAY(.1..40.) OF CHAR);
006C --       370         VAR J: 1..40; ERRPOS: 1..MAXLNL;
0000 0- A     371         BEGIN
000A --       372            IF LNPOS<>0 THEN ERRPOS:=LNPOS ELSE ERRPOS:=MAXLNL;
003C --       376         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
003C --       377            IF SWERRMSG THEN
004C 1-       378               BEGIN ERRCNT:=ERRCNT+1;
0064 --       379                     WRITE(ERRORS,' #K',CODE:3,SCNR:6,ERRPOS:4,' '); J:=1;
00C2 --       380                     WHILE J<40 DO
00CE 2-       381                        BEGIN WRITE(ERRORS,TEXT(.J.));
00FE --       382                           IF TEXT(.J.)=' '
0114 --       383                              THEN WHILE (TEXT(.J.)=' ') AND (J<40) DO J:=J+1
016A --       384                              ELSE J:=J+1
0190 -2       385                        END; WRITELN(ERRORS,TEXT(.40.));
01D0 --       386                     WRITELN(ERRORS,' ');
01EA 2-       387                     IF ERRCNT>=MAXNRERR THEN BEGIN WRITELN(ERRORS,
01FC -2       388                       '-***TOO MANY ERRORS IN THIS PROGRAM!***'); GOTO 9999 END
0224 -1       389               END
0224 -0 A     390         END;
027C --       391
027C -- A     392      PROCEDURE TOKENDESC(TNR: TOKENRG; VAR RETURNW: PACKED ARRAY(.1..10.) OF CHAR);
0000 0- A     393         BEGIN IF TNR<=WITHSYM THEN RETURNW:=KEYW(.TNR.)
0032 --       394                               ELSE RETURNW:=TOKW(.TNR.)
005E -0 A     395         END;
0084 --       396
0084 -- A     397      PROCEDURE ERRMSG26(WRONGTK: TOKENRG; TEXT: PACKED ARRAY(.1..17.) OF CHAR);
0000 0- A     398      BEGIN
000A --       399         ERRDESC:='1.......0......7 STARTS WITH 1.......0';
0016 --       400         FOR I:=1 TO 17 DO ERRDESC(.I.):=TEXT(.I.);
0092 --       401         TOKENDESC(WRONGTK,INSERTWORD);
00B8 --       402         FOR I:=1 TO 10 DO ERRDESC(.30+I.):=INSERTWORD(.I.);
0138 --       403         ERRMSG(26,ERRDESC)
014C -0 A     404      END;
01A0 --       405
01A0 -- A     406      PROCEDURE ERRMSG103(WRONGID: HASHPTR);
0000 0- A     407      BEGIN ERRDESC:='IDENTIFIER ''.........'' OF WRONG CLASS  ';
0016 --       408         FOR I:=1 TO 10 DO ERRDESC(.12+I.):=HASHNAME(.WRONGID.)(.I.);
00BA --       409         ERRMSG(103, ERRDESC)
```

```
00CE -0 A     410      END;
0124 --       411
0124 -- A     412      FUNCTION HASH: HASHPTR;
0048 --       413         VAR CURPOS: HASHPTR; MATCH: BOOLEAN;
0000 0- A     414         BEGIN
000A --       415            CURPOS:=(ORD(IDREG(.1.))+ORD(IDREG(.2.))+ORD(IDREG(.4.))
001C --       416                    +ORD(IDREG(.5.))) MOD NRBUCK + 1; (* HASHFUNCTION *)
0052 --       417            WHILE (HASHNAME(.CURPOS.)(.1.)<>'?') AND (HASHNAME(.CURPOS.)<>IDREG)
00BC --       418               DO CURPOS:=HASHNEXT(.CURPOS.);
0102 --       419            IF HASHNAME(.CURPOS.)(.1.)='?' THEN
0134 1-       420               BEGIN
0134 --       421                  IF FREESPACE=MAXNRID THEN
0146 2-       422                     BEGIN ERRDESC:='TOO MANY IDENTIFIERS                    ';
0152 --       423                           ERRMSG(249,ERRDESC); GOTO 9999
0178 -2       424                     END;
0178 --       425                  HASHNAME(.CURPOS.):=IDREG;
01A0 --       430                  (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
01A0 --       431                  FREESPACE:=FREESPACE+1; HASHNEXT(.CURPOS.):=FREESPACE;
020A -1       432               END;
020A --       433            HASH:=CURPOS
020A -0 A     434         END;
0274 --       435
0274 -- A     436      PROCEDURE NEXTCH;
0040 --       437
0040 -- B     438         PROCEDURE ENDOFLINE; VAR I: INTEGER;
0000 0- B     439            BEGIN
000A --       440                  WRITELN(OUTPUT,' ',STMNR:6,LEVEL:4,NEST:4,SCNR:6,'    ',LINE);
0094 --       441                  LINECNT;
0098 --       442                  FOR I:=1 TO MAXLNL DO LINE(.I.):=' ';
00DC --       443                  (* READ BLANK AFTER LAST CHARACTER *)
00DC --       444                  READ(INPUT,CH);
00EA --       445                  CH:=CHR(ORDEOLNCHR);
00F8 --       446                  SWEOLN:=FALSE; LNPOS:=0
00FE -0 B     447            END;
0124 --       448
0000 0- A     449         BEGIN
000A --       450            IF SWEOLN
000A --       451               THEN ENDOFLINE
0014 1-       452               ELSE BEGIN
001C --       453                       IF LNPOS=MAXLNL
001C 2-       454                          THEN BEGIN
0028 --       455                                  ERRDESC:='INPUTLINE EXCEEDS MAXIMAL LENGTH ALLOWED';
0034 --       456                                  ERRMSG(451,ERRDESC);
004C --       457                                  WHILE NOT EOLN(INPUT) DO READ(INPUT,CH);
0074 --       458                                  ENDOFLINE
0074 -2       459                               END
0078 2-       460                          ELSE BEGIN
007C --       461                                  LNPOS:=LNPOS+1;
009A --       462                                  IF NOT EOF(INPUT) THEN READ(INPUT,CH);
00BA --       463                                  IF EOF(INPUT) THEN
00C8 3-       464                                     BEGIN
00C8 --       465                                        ERRDESC:=
00C8 --       466                                           'END OF FILE FOUND BEFORE END OF PROGRAM ';
00D4 --       467                                        ERRMSG(450,ERRDESC); GOTO 9999
00FA -3       468                                     END;
00FA --       469                                  LINE(.LNPOS.):=CH;
011E --       470                                  SWEOLN:=EOLN(INPUT)
011E -2       471                               END
```

```
0140 -1       472                     END
0140 -0 A     473          END;
01A8 --       474
01A8 -- A     475       PROCEDURE ADVANCE; (* ASSIGNS NEXT TOKEN *)
0040 --       476          CONST NEXSTA=1; ACTION=2;
0040 --       477          VAR STATE: INTEGER; OLDCH: CHAR; I, J: 0..10;
0050 --       478              STRGLG: INTEGER;
0054 --       479              STRGEND, SWTRUNC: BOOLEAN;
005C --       480              CURCASE: DIGIT..OTHERCHR;
0060 --       481
0060 -- B     482          FUNCTION SRCHKW: ANDSYM..IDENTIFIER;
0048 --       483             VAR TOP, BOT, MID: INTEGER; NOTFND: BOOLEAN;
0000 0- B     484             BEGIN NOTFND:=TRUE; TOP:=ANDSYM; BOT:=WITHSYM;
0022 --       485                   WHILE (TOP<=BOT) AND NOTFND
0034 1-       486                      DO BEGIN MID:=(TOP+BOT) DIV 2;
0052 --       487                               IF IDREG<KEYW(.MID.)
006E --       488                                  THEN BOT:=MID-1
007A --       489                                  ELSE IF IDREG>KEYW(.MID.)
00A6 --       490                                          THEN TOP:=MID+1
00B2 --       491                                          ELSE NOTFND:=FALSE
00C2 -1       492                         END;
00CC --       493                    SRCHKW:=MID;
00E6 --       494                    IF  NOTFND THEN SRCHKW:=IDENTIFIER
00F0 -0 B     495             END;
0110 --       496
0110 -- B     497          FUNCTION COL: 1..6;  VAR ST: INTEGER;
0000 0- B     498             BEGIN ST:=CHRSEL(.CH.); IF ST>MINUS THEN COL:=5
003E --       499                                                 ELSE IF ST=MINUS THEN COL:=4
0056 --       500                                                                  ELSE COL:=ST
0062 -0 B     501             END;
0098 --       502
0098 --       503       (***************************)
0098 --       504       (* LEXICAL SCANNING STARTS *)
0098 --       505       (***************************)
0098 --       506
0000 0- A     507       BEGIN
000A --       508          TOKENNR:=0; TOKENFD:=0;
0022 1-       509          REPEAT (* UNTIL THERE IS A NEW TOKEN *)
0022 --       510             CURCASE:=CHRSEL(.CH.);
005C 2-       511             CASE CURCASE OF
0064 --       512
0064 --       513             DIGIT:
0064 3-       514                BEGIN STATE:=1; TOKENFD:=ORD(CH)-ORD('0');
0084 4-       515                   REPEAT NEXTCH;
0088 5-       516                          CASE TRANSIT(.STATE,COL,ACTION.) OF
00CC --       517                               1: TOKENFD:=10*TOKENFD+ORD(CH)-ORD('0');
00F6 --       518                               2: TOKENNR:=UNSGINTEG;
0108 6-       519                               3: BEGIN TOKENNR:=UNSGREAL;
0116 --       520                                     ERRDESC:=
0116 --       521                                        'REAL NUMBERS ARE NOT IMPLEMENTED        ';
0122 --       522                                     ERRMSG(398,ERRDESC)
0136 -6       523                                  END;
013E --       524                               4: ;
0142 --       525                               5: IF CH=')'
0142 6-       526                                     THEN BEGIN (* ')' AFTER INTEGER *)
0154 --       527                                             TOKENNR:=UNSGINTEG;
0162 --       528                                             CH:=']' (* RIGHT BRACKET *)
0162 -6       529                                          END
```

```
0170 --       530                                    ELSE IF CH='.'
0174 6-       531                                         THEN BEGIN (* '..' AFTER INTEGER *)
0186 --       532                                                 TOKENNR:=UNSGINTEG;
0194 --       533                                                 CH:=CHR(ORDDBLDOTCH)
0194 -6       534                                              END
01A2 6-       535                                         ELSE BEGIN TOKENNR:=UNSGREAL;
01B4 --       536                                                 ERRDESC:=
01B4 --       537                                           'ERROR IN REAL CONSTANT: DIGIT EXPECTED  ';
01C0 --       538                                                 ERRMSG(201,ERRDESC);
01D8 --       539                                                 ERRDESC:=
01D8 --       540                                           'REAL NUMBERS ARE NOT IMPLEMENTED        ';
01E4 --       541                                                 ERRMSG(398,ERRDESC)
01F8 -6       542                                              END;
0200 6-       543                               6: BEGIN
0200 --       544                                     ERRDESC:=
0200 --       545                                        'ERROR IN REAL CONSTANT: DIGIT EXPECTED  ';
020C --       546                                     ERRMSG(201,ERRDESC)
0220 -6       547                                  END
0224 -5       548                            END;
025A --       549                            STATE:=TRANSIT(.STATE,COL,NEXSTA.)
0296 -4       550                      UNTIL STATE=0
029E -3       551                   END;
02AE --       552
02AE --       553             PERIODCHR:
02AE 3-       554                BEGIN NEXTCH;
02B2 44       555                   IF CH='.' THEN BEGIN TOKENNR:=DOUBLEDOT; NEXTCH END
02D6 4-       556                            ELSE IF CH=')' THEN BEGIN TOKENNR:=RBRACKSYM;
02FA --       557                                                      NEXTCH
02FA -4       558                                                   END
02FE --       559                                ELSE TOKENNR:=PERIODSYM
0302 -3       560                END;
0314 --       561
0314 --       562             LETTER, ELETTER:
0314 3-       563                BEGIN IDREG:='          '; I:=0;
0320 4-       564                   REPEAT I:=I+1; IDREG(.I.):=CH; NEXTCH
0362 -4       565                   UNTIL (I=10) OR (CHRSEL(.CH.)<>LETTER) AND
03AA --       566                                   (CHRSEL(.CH.)<>ELETTER) AND (CHRSEL(.CH.)<>DIGIT);
041A --       567                   IF I=10 THEN
0426 4-       568                      BEGIN
0426 --       569                         SWTRUNC:=FALSE;
042C --       570                         WHILE (CHRSEL(.CH.)=LETTER) OR (CHRSEL(.CH.)=ELETTER)
0490 --       571                               OR (CHRSEL(.CH.)=DIGIT)
04C4 55       572                            DO BEGIN SWTRUNC:=TRUE; NEXTCH END;
04DC --       573                         IF SWTRUNC THEN
04E6 5-       574                            BEGIN
04E6 --       575                               ERRDESC:='IDENTIFIER TRUNCATED TO: .....      ';
04F2 --       576                               FOR J:=1 TO 10 DO ERRDESC(.25+J.):=IDREG(.J.);
056C --       577                               ERRMSG(398,ERRDESC)
0580 -5       578                            END
0584 -4       579                      END;
0584 --       580                   TOKENNR:=SRCHKW;
05A8 --       581                   IF TOKENNR=IDENTIFIER
05A8 --       582                      THEN TOKENFD:=HASH
05BA --       583                      ELSE IF TOKENNR IN (.ANDSYM, DIVSYM, INSYM, MODSYM.) THEN
060A 4-       584                           BEGIN IF TOKENNR=INSYM
060A 55       585                                 THEN BEGIN TOKENNR:=RELOPER; TOKENFD:=7 END
0638 5-       586                                 ELSE BEGIN CASE TOKENNR OF
064A --       587                                                 ANDSYM:TOKENFD:=5;
```

```
065C --      588                                          DIVSYM:TOKENFD:=3;
066E --      589                                          MODSYM:TOKENFD:=4
066E -6      590                                      END; TOKENNR:=MULTOPER
06E2 -5      591                                  END
06F0 -4      592                          END
06F0 -3      593              END;
06F4 --      594
06F4 --      595          PLUS,MINUS:
06F4 33      596             BEGIN TOKENNR:=PLUSMINUS; TOKENFD:=CURCASE-PLUS+1; NEXTCH END;
0720 --      597
0720 --      598          STAR,SLASH:
0720 3-      599             BEGIN TOKENNR:=MULTOPER; TOKENFD:=CURCASE-STAR+1; NEXTCH;
0748 --      600                IF TOKENFD=2 THEN
075A --      601                   ERRMSG(398, 'REAL NUMBERS ARE NOT IMPLEMENTED        ')
0768 -3      602             END;
0770 --      603
0770 --      604          LPARACHR:
0770 3-      605             BEGIN NEXTCH;
0774 --      606                IF CH='.'
0774 44      607                   THEN BEGIN TOKENNR:=LBRACKSYM; NEXTCH END
0798 --      608                   ELSE IF CH='*'
079C 4-      609                        THEN BEGIN
07AE --      610                                NEXTCH;
07B2 --      611                                (* CHECK FOR LIST OF COMPILE TIME OPTIONS *)
07B2 --      612                                (* S: NON STANDARD SYNTAX                 *)
07B2 --      613                                (* U: SUPERCRIPTS ON PAIRED PARENTHESIS   *)
07B2 --      614                                (* X: PRINTING OF X-REFERENCE LIST        *)
07B2 --      615                                IF CH='$' THEN
07C4 5-      616                                   REPEAT NEXTCH;
07C8 --      617                                      IF (CH='S') OR (CH='X') OR (CH='U') THEN
081A 6-      618                                         BEGIN OLDCH:=CH;
083A --      619                                            NEXTCH;
083E --      620                                            IF (CH='+') OR (CH='-') THEN
0876 7-      621                                               BEGIN
0876 8-      622                                                  CASE OLDCH OF
087E --      623                                                  'S': NOTSTAND:=(CH='+');
08B6 --      624                                                  'U': SUPERSCR:=(CH='+');
08EE --      625                                                  'X': XREFLIST:=(CH='+')
0906 -8      626                                                  END; (*CASE*)
0958 --      627                                                  NEXTCH
0958 -7      628                                               END
095C -6      629                                         END
095C -5      630                                   UNTIL CH<>',';
096E 5-      631                                REPEAT OLDCH:=CH; NEXTCH
098E -5      632                                UNTIL (OLDCH='*') AND (CH=')');
09C4 --      633                                NEXTCH
09C4 -4      634                             END
09C8 --      635                        ELSE TOKENNR:=LPARASYM
09CC -3      636             END;
09DE --      637
09DE --      638          RPARACHR, LBRACKCHR, RBRACKCHR,(*SEMICCHR,*)COMMACHR, AMPERSAND, EQUALCHR:
09DE 33      639             BEGIN TOKENNR:=CURCASE-RPARACHR+RPARASYM; NEXTCH END;
0A0E --      640
0A0E 33      641          SEMICCHR: BEGIN SCNR:=SCNR+1; TOKENNR:=SEMICSYM; NEXTCH END;
0A30 --      642
0A30 --      643
0A30 --      644          LESSCHR:
0A30 3-      645             BEGIN TOKENNR:=RELOPER; NEXTCH;
```

```
0A42 44      646                 IF CH='>' THEN BEGIN TOKENFD:=2; NEXTCH END
0A66 44      647                           ELSE IF CH='=' THEN BEGIN TOKENFD:=5; NEXTCH END
0A8E --      648                                         ELSE TOKENFD:=3
0A92 -3      649              END;
0AA4 --      650
0AA4 --      651           GREATRCHR:
0AA4 3-      652              BEGIN TOKENNR:=RELOPER; NEXTCH;
0AB6 44      653                 IF CH='=' THEN BEGIN TOKENFD:=6; NEXTCH END
0ADA --      654                           ELSE TOKENFD:=4
0ADE -3      655              END;
0AF0 --      656
0AF0 --      657           COLONCHR:
0AF0 3-      658              BEGIN NEXTCH;
0AF4 44      659                 IF CH='=' THEN BEGIN TOKENNR:=BECOMES; NEXTCH END
0B18 --      660                           ELSE TOKENNR:=COLONSYM
0B1C -3      661              END;
0B2E --      662
0B2E --      663           QUOTE:
0B2E --      664              (* TOKENFD DIV MAXSTRGL = POSITION OF FIRST CHARACTER IN STRING *)
0B2E --      665              (* TOKENFD MOD MAXSTRGL = LENGTH OF LITERAL SCANNED             *)
0B2E --      666              (* IF THE LITERAL EXTENDS OVER AN END-OF-LINE AN EOLN           *)
0B2E --      667              (* CHARACTER BE ASSUMED WITHIN IT.                              *)
0B2E 3-      668              BEGIN TOKENNR:=STRINGSYM; STRGLG:=STRGPTR; STRGEND:=FALSE;
0B50 --      669                 TOKENFD:=MAXSTRGL*STRGPTR;
0B68 --      670                 (*DE-1 WRITE(DEBUG,' STRINGCONSTANT FOUND:'); 1-BUG*)
0B68 4-      671                 REPEAT NEXTCH;
0B6C --      672                        IF CH=''''
0B6C 5-      673                           THEN BEGIN NEXTCH;
0B82 --      674                                      IF CH<>''''
0B82 6-      675                                         THEN BEGIN
0B94 --      676                                                 STRING(.STRGPTR.):=CHR(ORDEOSTRGCH);
0BB8 --      677                                                 STRGEND:=TRUE
0BB8 -6      678                                              END
0BC0 --      679                                         ELSE STRING(.STRGPTR.):=''''
0BE0 -5      680                                END
0BE8 --      681                           ELSE STRING(.STRGPTR.):=CH;
0C16 --      682                        IF STRGPTR=MAXSTRGL THEN
0C28 5-      683                           BEGIN ERRDESC:='TOO MANY LONG CONSTANTS IN THIS PROGRAM ';
0C34 --      684                                 ERRMSG(254,ERRDESC); GOTO 9999
0C5A -5      685                           END;
0C5A --      686                        (*DE-1 WRITE(DEBUG,STRING(.STRGPTR.)); 1-BUG*)
0C5A --      687                        STRGPTR:=STRGPTR+1
0C5A -4      688                 UNTIL STRGEND;
0C8E --      689                 STRGLG:=STRGPTR-STRGLG-1; TOKENFD:=TOKENFD+STRGLG
0CA4 --      690              (*DE-1;WRITELN(DEBUG,' ') 1-BUG*)
0CA4 -3      691              END;
0CC0 --      692
0CC0 33      693           BLANK: REPEAT NEXTCH UNTIL CH<>' ';
0CDA --      694
0CDA --      695           EOLNCHR: NEXTCH;
0CE2 --      696
0CE2 --      697           LBRACECHR:
0CE2 33      698              BEGIN REPEAT NEXTCH UNTIL CH='}'; NEXTCH END;
0D00 --      699
0D00 --      700           DBLDOTCHR:
0D00 33      701              BEGIN TOKENNR:=DOUBLEDOT; NEXTCH END;
0D16 --      702
0D16 --      703           OTHERCHR:
```

```
0D16 3-       704                BEGIN ERRDESC:='ILLEGAL CHARACTER ''.'' IN PROGRAMTEXT    ';
0D22 --       705                      ERRDESC(.20.):=CH; ERRMSG(6,ERRDESC); NEXTCH
0D4E -3       706                END
0D52 --       707
0D52 -2       708             END (* OF CASE *)
0DD4 -1       709          UNTIL TOKENNR<>0;
0DE6 --       714          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0DE6 -0 A     715       END; (* OF ADVANCE *)
0F64 --       716
0F64 --       717       (*********************************************************)
0F64 --       718       (* FOLLOWING ARE THE STRING MANIPULATING ROUTINES:       *)
0F64 --       719       (* COMPACT, LOC, FREE, SSS, APP, CAT, LIT, WRAP, OUT     *)
0F64 --       720       (*********************************************************)
0F64 --       721
0F64 --       722       (* GENERALLY ALL BUT LOC ARE DESIGNED THAT THERE ARGUMENTS ARE ASSUMED  *)
0F64 --       723       (* TO BE LOCATED ALREADY. IF DURING OPERATION A STRING ALLOCATION IS    *)
0F64 --       724       (* TOO SHORT TO HOLD A NEW RESULT, IT WILL BE REALLOCATED TO THE NEEDED *)
0F64 --       725       (* SIZE. THIS ENABLES THE COMPACTIFICATION ROUTINE TO COMPRESS STRINGS  *)
0F64 --       726       (* TO THEIR CURRENT SIZE IF EXTREME SHORTAGE OF WORKSPACE ARISES.       *)
0F64 --       727
0F64 -- A     728       PROCEDURE OUT(STRG: CHARVAR);
0044 --       729       (* WRITES A STRING ONTO SPUNCH, MAXCODLNL CHARACTERS PER LINE *)
0044 --       730       VAR CURPOS: INTEGER;
0000 0- A     731       BEGIN
000A --       732          CURPOS:=0;
0010 --       733          WHILE CURPOS<STRG@.CURLG DO
0030 1-       734             BEGIN
0030 --       735                FOR I:=1 TO MAXCODLNL DO
0036 --       736                    IF CURPOS<STRG@.CURLG
005C 2-       737                       THEN BEGIN
0068 --       738                               WRITE(SPUNCH, WORKSPACE(.STRG@.WSPOS+CURPOS.));
00B6 --       739                               CURPOS:=CURPOS+1
00B6 -2       740                            END;
00DC --       741                WRITELN(SPUNCH);
00E6 -1       742             END;
00EA --       743          WRITELN(SPUNCH,'*')
0100 -0 A     744       END; (*OUT*)
0124 --       745
0124 -- A     746       PROCEDURE COMPACT(VAR RSTRG: CHARVAR; SIZE: INTEGER);
0048 --       747       (* SHALL RETURN A STRING OF SIZE WHICH COULD NOT BE ALLOCATED *)
0048 --       748       (* THEREBY IT COMPRESSES THE WORKSPACE AS MUCH AS POSSIBLE    *)
0048 --       749       (* MAXLG'S MAY BE TRUNCATED TO CURLG'S IF NEEDED              *)
0048 --       750       VAR TRAV, TAIL, SAVE: @STRGDESC; OLDP: 1..MAXWSL;
0058 --       751           USED: INTEGER; (* CURRENTLY USED SPACE *)
005C --       752           BEYOND: INTEGER; (* SPACE NEEDED BEYOND UNUSED ONE *)
0060 --       753       (*DE-4 OLDREMWS, CURPOS: INTEGER; 4-BUG*)
0000 0- A     754       BEGIN
000A --       755          (*DE-4 OLDREMWS:=REMWS; 4-BUG*)
000A --       756          (* SORT OUT ALL UNUSED STRINGS FROM ALLOCS *)
000A --       757          TRAV:=ALLOCS; ALLOCS:=NIL; TAIL:=NIL; USED:=0;
0046 --       758          WHILE TRAV<>NIL DO
0052 --       759             IF TRAV@.MAXLG>0
0066 1-       760                THEN BEGIN
0072 --       761                        USED:=USED+TRAV@.MAXLG;
0092 --       762                        (* PUT IT ONTO NEW ALLOCS *)
0092 --       763                        IF TAIL=NIL
0092 --       764                           THEN ALLOCS:=TRAV
009E --       765                           ELSE TAIL@.NXTSTRG:=TRAV;
```

```
00F8 --     766                     TAIL:=TRAV; TRAV:=TRAV@.NXTSTRG
012A -1     767                   END
0148 1-     768               ELSE BEGIN (* PUT IT AWAY *)
014C --     769                     SAVE:=TRAV; TRAV:=TRAV@.NXTSTRG;
019C --     770                     SAVE@.NXTSTRG:=EXTRAS; EXTRAS:=SAVE
01D4 -1     771                   END;
01FC --     772         IF TAIL<>NIL THEN TAIL@.NXTSTRG:=NIL;
0222 --     773         BEYOND:=USED+SIZE-MAXWSL;
0232 --     778         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0232 --     779         (* COMPACTIFICATION *)
0232 --     780         TRAV:=ALLOCS; REMWS:=1;
0264 --     781         WHILE TRAV<>NIL DO
0270 1-     782            BEGIN
0270 --     783               (* DON'T MOVE UNNESSESARY ONES *)
0270 --     784               IF TRAV@.WSPOS=REMWS
0284 --     785                  THEN IF BEYOND>0
0296 2-     786                          THEN BEGIN REMWS:=REMWS+TRAV@.CURLG;
02CE --     787                                     BEYOND:=BEYOND-TRAV@.MAXLG+TRAV@.CURLG;
0306 --     788                                     TRAV@.MAXLG:=TRAV@.CURLG
032E -2     789                               END
0336 --     790                          ELSE REMWS:=REMWS+TRAV@.MAXLG
034E 2-     791                  ELSE BEGIN
036A --     792                          OLDP:=TRAV@.WSPOS;
0398 --     793                          TRAV@.WSPOS:=REMWS;
03CC --     794                          IF TRAV@.CURLG<>0 THEN
03EC --     795                             FOR I:=0 TO TRAV@.CURLG-1 DO
0410 3-     796                                 BEGIN WORKSPACE(.REMWS.):=WORKSPACE(.OLDP+I.);
0476 -3     797                                       REMWS:=REMWS+1 END;
04A8 --     798                          IF BEYOND>0
04A8 3-     799                             THEN BEGIN
04B4 --     800                                     BEYOND:=BEYOND-TRAV@.MAXLG+TRAV@.CURLG;
04EC --     801                                     TRAV@.MAXLG:=TRAV@.CURLG
0514 -3     802                                END
051C --     803                             ELSE REMWS:=REMWS+TRAV@.MAXLG-TRAV@.CURLG;
0564 -2     804                       END;
0564 --     805                  TRAV:=TRAV@.NXTSTRG
0578 -1     806            END;
059A --     807         (* TRY TO SATISFY REQUEST NOW *)
059A --     808         IF MAXWSL-REMWS+1<SIZE
05AC 1-     809            THEN BEGIN ERRMSG(400, 'STRING-WORKSPACE OVERFLOW                ');
05C6 --     831                       (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
05C6 -1     832                       GOTO 9999 END
05D4 1-     833            ELSE BEGIN
05D8 --     834                    IF EXTRAS<>NIL
05D8 22     835                       THEN BEGIN RSTRG:=EXTRAS; EXTRAS:=EXTRAS@.NXTSTRG END
0650 --     836                       ELSE NEW(RSTRG);
0668 --     837                    IF TAIL=NIL
0668 --     838                       THEN ALLOCS:=RSTRG
0674 --     839                       ELSE TAIL@.NXTSTRG:=RSTRG;
06D6 --     840                    RSTRG@.NXTSTRG:=NIL;
06F4 --     841                    RSTRG@.WSPOS:=REMWS; RSTRG@.MAXLG:=SIZE;
074C --     842                    RSTRG@.CURLG:=1; WORKSPACE(.REMWS.):='#';
0796 --     843                    REMWS:=REMWS+SIZE
0796 -1     844                 END
07AE --     849         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
07AE -0 A   850      END; (* COMPACT*)
07F8 --     851
07F8 -- A   852      PROCEDURE LOC(VAR RSTRG: CHARVAR; SIZE: INTEGER);
```

```
0048 --      853      (* ALLOCATES STRING OF SIZE IN WORKSPACE, SETS IT '#' *)
0048 --      854      VAR TRAV, TAIL: @STRGDESC;
0000 0- A    855      BEGIN
000A --      856         (* ALL STRINGS CONTAIN AT LEAST ONE CHARACTER *)
000A --      857         IF SIZE=0 THEN SIZE:=1;
001E --      858         (* IS THERE AN UNUSED STRING LARGE ENOUGH? *)
001E --      859         TRAV:=ALLOCS; TAIL:=NIL; RSTRG:=NIL;
0052 --      860         WHILE (TRAV<>NIL) AND (RSTRG=NIL) DO
0082 --      861            IF (-TRAV@.MAXLG)>=SIZE
009C --      862               THEN RSTRG:=TRAV
00A4 11      863               ELSE BEGIN TAIL:=TRAV; TRAV:=TRAV@.NXTSTRG END;
011E --      864         (* IF YES, TAKE IT *)
011E --      865         IF RSTRG<>NIL
011E 1-      866            THEN BEGIN RSTRG@.MAXLG:=SIZE;
014E -1      867                    RSTRG@.CURLG:=1; WORKSPACE(.RSTRG@.WSPOS.):='#' END
01AA --      868            ELSE IF MAXWSL-REMWS+1>=SIZE
01C0 1-      869                    THEN BEGIN
01C8 --      870                           IF EXTRAS<>NIL
01C8 22      871                              THEN BEGIN RSTRG:=EXTRAS; EXTRAS:=EXTRAS@.NXTSTRG END
0240 --      872                              ELSE NEW(RSTRG);
0258 --      873                           IF TAIL=NIL
0258 --      874                              THEN ALLOCS:=RSTRG
0264 --      875                              ELSE TAIL@.NXTSTRG:=RSTRG;
02C6 --      876                         RSTRG@.NXTSTRG:=NIL;
02E4 --      877                         RSTRG@.WSPOS:=REMWS; RSTRG@.MAXLG:=SIZE;
033C --      878                         RSTRG@.CURLG:=1; WORKSPACE(.REMWS.):='#';
0386 --      879                         REMWS:=REMWS+SIZE
0386 -1      880                      END
039E --      881                    ELSE COMPACT(RSTRG, SIZE)
03B2 -0 A    882      END; (* LOC *)
03D4 --      883
03D4 -- A    884      PROCEDURE FREE(UNUSED: CHARVAR);
0044 --      885      (* MARKS STRING FOR REUSE OR REMOVAL *)
0000 0- A    886      BEGIN
000A --      887         UNUSED@.MAXLG:=-ABS(UNUSED@.MAXLG)
0038 -0 A    888      END; (*FREE*)
0044 --      889
0044 -- A    890      PROCEDURE SSS(VAR TARGET: CHARVAR; SOURCE: CHARVAR);
0048 --      891      (* ASSIGNS SOURCESTRING TO TARGETSTRING *)
0000 0- A    892      BEGIN
000A --      893         IF TARGET@.MAXLG<SOURCE@.CURLG
0036 1-      894            THEN BEGIN
0042 --      898                    (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0042 --      899                    FREE(TARGET);
0068 --      900                    LOC(TARGET, SOURCE@.CURLG)
008C -1      901                 END;
0090 --      902         TARGET@.CURLG:=SOURCE@.CURLG;
00C4 --      903         IF SOURCE@.CURLG>0 THEN
00E4 --      904            FOR I:=0 TO SOURCE@.CURLG-1 DO
0108 --      905               WORKSPACE(.TARGET@.WSPOS+I.):=WORKSPACE(.SOURCE@.WSPOS+I.)
0186 -0 A    906      END; (*SSS*)
01D4 --      907
01D4 -- A    908      PROCEDURE APP(VAR LEFT: CHARVAR; RIGHT: CHARVAR);
0048 --      909      (* APPENDS SECOND STRING TO FIRST ONE *)
0048 --      910      VAR AUXIL: @STRGDESC;
0000 0- A    911      BEGIN
000A --      912         IF LEFT@.CURLG+RIGHT@.CURLG>LEFT@.MAXLG
0056 1-      913            THEN BEGIN
```

```
005E --       917                     (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
005E --       918                     LOC(AUXIL, LEFT@.CURLG+RIGHT@.CURLG);
00A2 --       919                     SSS(AUXIL, LEFT);
00D0 --       920                     FREE(LEFT); LEFT:=AUXIL
00F6 -1       921                  END;
0118 --       922            IF RIGHT@.CURLG<>0
012C 1-       923               THEN BEGIN
0138 --       924                     FOR I:=0 TO RIGHT@.CURLG-1 DO
015C --       925                         WORKSPACE(.LEFT@.WSPOS+LEFT@.CURLG+I.):=
01C2 --       926                                                 WORKSPACE(.RIGHT@.WSPOS+I.);
0228 --       927                     LEFT@.CURLG:=LEFT@.CURLG+RIGHT@.CURLG
026C -1       928                  END
0278 -0 A     929         END; (*APP*)
0294 --       930
0294 -- A     931         PROCEDURE CAT(VAR TARGET: CHARVAR; LEFT, RIGHT: CHARVAR);
004C --       932         (* CONCATENATES LEFT AND RIGHT AND ASSIGNS RESULT TO TARGET   *)
004C --       933         VAR AUXIL: @STRGDESC;
0000 0- A     934         BEGIN
000A --       935            (* CHECK IF RESULT IS RIGHT ARGUMENT *)
000A --       936            IF TARGET@.WSPOS=RIGHT@.WSPOS
0036 11       937               THEN BEGIN LOC(AUXIL, RIGHT@.CURLG); SSS(AUXIL, RIGHT) END
0094 --       938               ELSE AUXIL:=NIL;
009E --       939            SSS(TARGET, LEFT);
00C8 --       940            IF AUXIL=NIL
00C8 --       941               THEN APP(TARGET, RIGHT)
00FA 11       942               ELSE BEGIN APP(TARGET, AUXIL); FREE(AUXIL) END
014E -0 A     943         END; (*CAT*)
0158 --       944
0158 -- A     945         PROCEDURE LIT(VAR TARGET: CHARVAR; SIZE: INTEGER;
0040 --       946                       LITERAL: PACKED ARRAY(.1..15.) OF CHAR);
0057 --       947         (* SETS TARGETSTRING TO THE FIRST SIZE CHARACTERS OF LITERAL  *)
0000 0- A     948         BEGIN
000A --       949            IF SIZE>TARGET@.MAXLG
0022 1-       950               THEN BEGIN
002E --       954                     (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
002E --       955                     FREE(TARGET);
0054 --       956                     LOC(TARGET, SIZE)
0064 -1       957                  END;
0068 --       958            TARGET@.CURLG:=SIZE;
0088 --       959            IF SIZE<>0 THEN
0094 --       960               FOR I:=1 TO SIZE DO
00A2 --       961                  WORKSPACE(.TARGET@.WSPOS+I-1.):=LITERAL(.I.)
010C -0 A     962         END; (*LIT*)
0158 --       963
0158 -- A     964         PROCEDURE WRAPNRS(VAR PIECE: CHARVAR); FORWARD; (* OH PASCAL! *)
0044 --       965
0044 -- A     966         PROCEDURE WRAP(CH1: CHAR; VAR PIECE: CHARVAR; CH2: CHAR);
004C --       967         (* PUTS CH1 IN FRONT AND CH2 IN THE END OF PIECE *)
004C --       968         (* MAY USE WRAPNRS ON CERTAIN CONDITIONS INSTEAD *)
004C --       969         VAR AUXIL: @STRGDESC;
0000 0- A     970         BEGIN
000A --       971            (* SUPERSCRIPTS WANTED? -- NOT PART OF STRING SYSTEM -- *)
000A 1-       972            IF (CH1='(') AND (CH2=')') AND SUPERSCR THEN WRAPNRS(PIECE) ELSE BEGIN
0050 --       973            IF PIECE@.CURLG+2>PIECE@.MAXLG
0088 2-       974               THEN BEGIN
0090 --       978                     (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0090 --       979                     LOC(AUXIL, PIECE@.CURLG+2);
00C0 --       980                     SSS(AUXIL, PIECE);
```

```
00EE --   981                     FREE(PIECE); PIECE:=AUXIL
0114 -2   982                   END;
0136 --   983           FOR I:=PIECE@.CURLG DOWNTO 1 DO
0154 --   984               WORKSPACE(.PIECE@.WSPOS+I.):=WORKSPACE(.PIECE@.WSPOS+I-1.);
020A --   985           WORKSPACE(.PIECE@.WSPOS.):=CH1;
0246 --   986           WORKSPACE(.PIECE@.WSPOS+PIECE@.CURLG+1.):=CH2;
02A2 --   987           PIECE@.CURLG:=PIECE@.CURLG+2
02D2 -1   988           END (* OF NO SUPERSCR *)
02DE -0 A 989       END; (*WRAP*)
0304 --   990
0304 --   991       (*****************************)
0304 --   992       (* BEGIN OF PARSING ROUTINES *)
0304 --   993       (*****************************)
0304 --   994
0304 -- A 995       PROCEDURE SYNCHRONIZE;
0000 0- A 996          BEGIN WHILE NOT(TOKENNR IN SYNCHRSYM)
0050 --   997                      DO IF TOKENNR IN DONTSKIPSYM
004A --   998                         THEN GOTO 9999
0098 --   999                         ELSE ADVANCE
009C -0 A 1000         END;
00B4 --   1001
00B4 -- A 1002      PROCEDURE TERMINAL(MATCHNR: TOKENRG; VAR RETNR: INTEGER);
0000 0- A 1003         BEGIN
000A --   1004            IF NOTSKIP
000A --   1005               THEN IF MATCHNR=TOKENNR
001A 1-   1006                    THEN BEGIN
002C --   1011                            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
002C --   1012                            RETNR:=TOKENFD; ADVANCE
003E -1   1013                       END
0042 1-   1014                    ELSE BEGIN
0046 --   1015                            ERRDESC:='.......... EXPECTED,BUT .......... FOUND';
0052 --   1016                            TOKENDESC(MATCHNR,INSERTWORD);
0078 --   1017                            FOR I:=1 TO 10 DO ERRDESC(.I.):=INSERTWORD(.I.);
00F4 --   1018                            TOKENDESC(TOKENNR,INSERTWORD);
0120 --   1019                            FOR I:=1 TO 10 DO ERRDESC(.24+I.):=INSERTWORD(.I.);
01A0 --   1020                            ERRMSG(25,ERRDESC);
01B8 --   1021                            SYNCHRONIZE;
01BC --   1022                            IF MATCHNR<>TOKENNR
01BC 22   1023                               THEN BEGIN NOTSKIP:=FALSE; RETNR:=0 END
01E4 --   1024                               ELSE RETNR:=TOKENFD
01E8 -1   1025                       END
01FA --   1026               ELSE IF (MATCHNR IN SYNCHRSYM) AND (MATCHNR=TOKENNR)
0256 11   1027                    THEN BEGIN RETNR:=TOKENFD; ADVANCE; NOTSKIP:=TRUE END
0282 --   1028                    ELSE RETNR:=0
0286 -0 A 1029         END;
02E0 --   1030
02E0 -- A 1031      FUNCTION CONVPSID(IDVAL: INTEGER;
0048 --   1032                        PSNAME: PACKED ARRAY(.1..5.) OF CHAR): HASHPTR;
0051 --   1033      (* CATENATES '$', PSNAME AND THE LAST 4 DIGITS OF IDVAL TO FORM A PSEUDO- *)
0051 --   1034      (* IDENTIFIER. THEN RETURNS THE HASH ADDRESS OF IT. USED FOR LABELS AND   *)
0051 --   1035      (* IMPLICIT SCALAR TYPES.                                                 *)
0051 --   1036         VAR SPELLED: PACKED ARRAY(.0..10.) OF CHAR; LN: INTEGER;
0000 0- A 1037         BEGIN IF NOTSKIP
000A 1-   1038            THEN BEGIN LN:=CONV(IDVAL,SPELLED);
0036 --   1039                 IDREG:='$       ';
003C --   1040                 FOR I:=1 TO 5 DO IDREG(.1+I.):=PSNAME(.I.);
00B6 --   1041                 IF LN>4 THEN
00C2 2-   1042                    BEGIN ERRDESC:='..... TRUNCATED TO ....                ';
```

```
00CE --      1043                          FOR I:=1 TO 5 DO ERRDESC(.I.):=PSNAME(.I.);
014A --      1044                          ERRMSG(398,ERRDESC); LN:=4
0162 -2      1045                       END;
016A --      1046                    FOR I:=1 TO LN DO IDREG(.6+I.):=SPELLED(.I.);
01EC --      1047                    CONVPSID:=HASH
01EC -1      1048                 END
020A --      1049              ELSE CONVPSID:=0
020E -0 A    1050           END; (* CONVPSID *)
0274 --      1051
0274 -- A   1052        FUNCTION BLDLAB(LABNR: HASHPTR): @ACTIVS;
004C --      1053        (* CREATES AN IDENTIFIER RECORD FOR A LABEL *)
004C --      1054           VAR RETACT: @ACTIVS;
0000 0- A    1055           BEGIN NEW(RETACT,SCALID); NEW(RETACT@.STYP,SCALTYP);
004A --      1056              WITH RETACT@, STYP@ DO
0076 1-      1057                 BEGIN IDKLASS:=SCALID; TYPKLASS:=SCALTYP;
0082 --      1058                    IDNR:=LABNR; LEVNR:=LEVEL;
00AA --      1059                    SCIDNR:=SPLABEL; SCLEVNR:=LEVEL;
00D8 --      1060                    LOWBND:=0; HIGBND:=9999; VALUU:=0;
00EC --      1061                    NEW(XREF); XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL
012C -1      1062                 END;
0132 --      1063              BLDLAB:=RETACT
0132 -0 A    1064           END; (* BLDLAB *)
0170 --      1065
0170 -- A   1066        FUNCTION BLDSTRG(ITSFIELD: INTEGER): @TYPES;
004C --      1067        (* CREATES A TYPE FOR A LITERAL STRING WITH MORE THAN ONE CHARACTER *)
004C --      1068           VAR RETTYP: @TYPES;
0000 0- A    1069           BEGIN NEW(RETTYP, ARRTYP);
0022 --      1070              WITH RETTYP@ DO
0038 1-      1071                 BEGIN SWPACKA:=TRUE;
0040 --      1072                    NEW(INDXTYP, SCALTYP);
0056 --      1073                    INDXTYP@.SCIDNR:=SPINTEGER;
008A --      1074                    INDXTYP@.SCLEVNR:=0;
00A4 --      1075                    INDXTYP@.LOWBND:=1;
00C0 --      1076                    INDXTYP@.HIGBND:=ITSFIELD MOD MAXSTRGL;
00E4 --      1077                    COMPTYP:=TYPCHAR
00E4 -1      1078                 END;
0108 --      1079              BLDSTRG:=RETTYP
0108 -0 A    1080           END; (* BLDSTRG *)
0140 --      1081
0140 -- A   1082        FUNCTION SEARCHONLY(ID: HASHPTR): BOOLEAN;
004C --      1083        (* CHECK IF IDENTIFIER ALREADY DECLARED *)
004C --      1084        LABEL 1; VAR PTRACT: @ACTIVS; NEWX:@XREFS; SWFND: BOOLEAN;
0000 0- A    1085        BEGIN
000A --      1086           IF NOTSKIP
000A 1-      1087           THEN BEGIN
001A --      1088                   SWFND:=FALSE;
0020 --      1089                   IF SRCHTAB(.ID.)<>NIL THEN
004C --      1090                      SWFND:=(SRCHTAB(.ID.)@.ACTLSTPTR@.LEVNR>=LEVEL);
00C2 --      1091                   IF SWFND
00C2 2-      1092                      THEN BEGIN
00CC --      1093                              PTRACT:=SRCHTAB(.ID.)@.ACTLSTPTR;
011E --      1094                              (* SUPPRESS ERROR MESSAGE IN FORWARD REFERENCE *)
011E --      1095                              IF PTRACT@.IDKLASS IN (.PROCID,FUNCID.) THEN
0166 --      1096                                 IF PTRACT@.SWFORW THEN GOTO 1;
0188 --      1097                              INSERTWORD:=HASHNAME(.ID.);
01B0 --      1098                              ERRDESC:='IDENTIFIER ''          '' DECLARED TWICE  ';
01BC --      1099                              FOR I:=1 TO 10 DO ERRDESC(.12+I.):=INSERTWORD(.I.);
023C --      1100                              ERRMSG(101,ERRDESC);
```

```
0254 --      1101                            1:NEW(NEWX); NEWX@.OCC:=SCNR;
0280 --      1102                              NEWX@.NXTREF:=PTRACT@.XREF; PTRACT@.XREF:=NEWX;
02F8 --      1103                              SEARCHONLY:=FALSE;
02FE -2      1104                           END
02FE --      1105                     ELSE SEARCHONLY:=TRUE
0302 -1      1106                END
030A --      1107             ELSE SEARCHONLY:=FALSE (* RETURN SOMETHING IN ERRORMODE *)
030E -0 A    1108        END; (* OF SEARCHONLY *)
0378 --      1109
0378 -- A    1110        PROCEDURE PRINTTYP(WHICHTYP: @TYPES);
0044 --      1111           VAR PTRACT: @ACTIVS;
0000 0- A    1112           BEGIN WITH WHICHTYP@ DO
0020 1-      1113              CASE TYPKLASS OF
0028 2-      1114                 SCALTYP: BEGIN
0028 --      1115                             WRITELN(XRFF,HASHNAME(.SCIDNR.),' ');
0070 --      1116                             IF LOWBND<=HIGBND THEN
007C --      1117                                WRITELN(XRFF,LOWBND,' ..',HIGBND,' (ORDERS ONLY) ')
00C2 -2      1118                          END;
00CA 2-      1119                 ARRTYP: BEGIN
00CA --      1120                             IF SWPACKA THEN WRITELN(XRFF,'PACKED ');
00EE --      1121                             WRITELN(XRFF,'ARRAY (.');
0108 --      1122                             PRINTTYP(INDXTYP); WRITELN(XRFF,'.) OF ');
0144 --      1123                             PRINTTYP(COMPTYP);
0166 -2      1124                          END;
016A 2-      1125                 RECTYP: BEGIN
016A --      1126                             IF SWPACKR THEN WRITELN(XRFF,'PACKED ');
018E --      1127                             WRITELN(XRFF,'RECORD ');
01A8 --      1128                             PTRACT:=SECTNS;
01C6 --      1129                             WHILE PTRACT<>NIL DO
01D2 3-      1130                                BEGIN WRITELN(XRFF,HASHNAME(.PTRACT@.IDNR.),':, ');
022E -3      1131                                      PTRACT:=PTRACT@.NXTACT END
0260 -2      1132                          END;
0268 --      1133                 PTRTYP: WRITELN(XRFF,'@',HASHNAME(.PTRIDNR.));
02B4 2-      1134                 UNDEF: BEGIN
02B4 --      1135                           IF UNDFIDNR<>0 THEN
02C0 --      1136                              WRITELN(XRFF,HASHNAME(.UNDFIDNR.),', ');
0308 --      1137                           WRITELN(XRFF,'*** UNDEFINED ***')
031E -2      1138                        END
0322 -1      1139                 END (* OF CASE *)
0354 -0 A    1140           END; (* OF PRINTTYP *)
03C4 --      1141
03C4 -- A    1142        PROCEDURE PRINTXREF(ACT: @ACTIVS);
0044 --      1143           LABEL 1;
0044 --      1144           VAR PTRARG: @ARGS; OLDER, OLD, NEW: @XREFS;
0000 0- A    1145           BEGIN REWRITE(XRFF);
0014 --      1146              WITH ACT@ DO
002A 1-      1147                 BEGIN (* REVERSE CROSS REFERENCE LIST *)
002A --      1148                    OLD:=NIL; NEW:=XREF;
004E --      1149                    WHILE NEW<>NIL DO
005A 2-      1150                       BEGIN OLDER:=OLD; OLD:=NEW; NEW:=NEW@.NXTREF;
00C8 -2      1151                          OLD@.NXTREF:=OLDER END;
00FE --      1152                    (* OLD KEEPS HEAD OF LIST *)
00FE --      1153                    IF OLD=NIL THEN GOTO 1; (* DON'T PRINT UNREF. NAMES *)
010E --      1154                    WRITELN(XRFF,OLD@.OCC:6,' ',HASHNAME(.IDNR.),' ');
018A 2-      1155                    CASE IDKLASS OF
0192 33      1156                    SCALID: BEGIN WRITELN(XRFF,'SCALAR, '); PRINTTYP(STYP) END;
01D2 33      1157                    TYPID: BEGIN WRITELN(XRFF,'TYPE, '); PRINTTYP(TYP) END;
0212 33      1158                    VARID: BEGIN WRITELN(XRFF,'VARIABLE, '); PRINTTYP(TYP) END;
```

```
0252 33      1159                  FLDID: BEGIN WRITELN(XRFF,'RECORD FIELD, '); PRINTTYP(TYP) END;
0292 --      1160                  PROCID, FUNCID:
0292 3-      1161                     BEGIN WRITELN(XRFF,'ENTRY('); PTRARG:=ARG;
02CA --      1162                        WHILE PTRARG<>NIL DO
02D6 4-      1163                           BEGIN IF PTRARG@.PASSKND THEN WRITELN(XRFF,'VAR, ');
030E --      1164                              PRINTTYP(PTRARG@.ARGTYP); WRITELN(XRFF,'; ');
035E --      1165                              PTRARG:=PTRARG@.NXTARG
0372 -4      1166                           END;
0394 --      1167                        WRITELN(XRFF,')');
03AE --      1168                        IF IDKLASS=FUNCID THEN
03BA 44      1169                           BEGIN WRITELN(XRFF,': '); PRINTTYP(RETTYP) END;
03F6 --      1170                        IF SWFORW THEN
0400 4-      1171                           BEGIN
0400 --      1172                              ERRDESC:='UNSATISF. FORWARD REFERENCE ''.........''';
040C --      1173                              INSERTWORD:=HASHNAME(.IDNR.);
0434 --      1174                              FOR I:=1 TO 10 DO ERRDESC(.29+I.):=INSERTWORD(.I.);
04B4 --      1175                              ERRMSG(117,ERRDESC);
04CC --      1176                              WRITELN(XRFF,'*** UNRESOLVED FORWARD REFERENCE ***')
04E2 -4      1177                           END
04E6 -3      1178                     END
04E6 -2      1179                  END (* OF CASE *)
051C -1      1180               END; (* OF WITH *)
051C --      1181            NEW:=OLD@.NXTREF;
054E --      1182            WHILE NEW<>NIL DO
055A 11      1183               BEGIN WRITELN(XRFF,NEW@.OCC:6,', '); NEW:=NEW@.NXTREF END;
05CE --      1184            1:WRITELN(XRFF,CHR(ORDEOSTRGCH)); RESET(XRFF);
05F2 -0 A    1185         END; (* OF PRINTXREF *)
06C0 --      1186
06C0 -- A    1187      PROCEDURE PUSH(NEWACT: @ACTIVS);
0044 --      1188         VAR ACTLSTEL: @ACTLST;
0048 --      1189         (*DE-3 I: INTEGER; 3-BUG*)
0000 0- A    1190         BEGIN IF NEWACT<>NIL
000A 1-      1191            THEN BEGIN
0016 --      1192                    NEWACT@.NXTACT:=STACK; STACK:=NEWACT;
0072 --      1193                    NEW(ACTLSTEL);
0082 --      1194                    ACTLSTEL@.ACTLSTPTR:=NEWACT;
00B4 --      1195                    ACTLSTEL@.NXTONE:=SRCHTAB(.NEWACT@.IDNR.);
011A --      1196                    SRCHTAB(.NEWACT@.IDNR.):=ACTLSTEL
0148 --      1211                    (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0148 -1      1212                 END
016C --      1213            (*DE-3 ELSE WRITELN(DEBUG,' *** ATTEMPT TO PUSH NIL ***') 3-BUG*)
016C -0 A    1214         END; (* OF PUSH *)
0184 --      1215
0184 -- A    1216      PROCEDURE POP;
0040 --      1217         LABEL 1;
0040 --      1218         VAR LINK, NEWLINK: @ACTIVS; PTR: @ACTLST;
004C --      1219             LABWORD: PACKED ARRAY(.1..5.) OF CHAR;
0051 --      1220
0051 -- B    1221         PROCEDURE SORT(SORTIN: @ACTIVS);
0044 --      1222         (* SORT INTO ATTRIBUTELIST *)
0044 --      1223         VAR OLD, NEW: @ACTIVS; NOTFND: BOOLEAN;
0000 0- B    1224         BEGIN
000A --      1228            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1229            OLD:=NIL; NEW:=ATTR;
0034 --      1230            IF NEW<>NIL
0034 1-      1231               THEN BEGIN NOTFND:=TRUE;
0048 --      1232                       WHILE (NEW<>NIL) AND NOTFND DO
0064 --      1233                          IF HASHNAME(.SORTIN@.IDNR.)<HASHNAME(.NEW@.IDNR.)
```

```
00C4 --      1234                            THEN NOTFND:=FALSE
00DA 22      1235                            ELSE BEGIN OLD:=NEW; NEW:=OLD@.NXTACT END;
0138 --      1236                       IF OLD<>NIL THEN OLD@.NXTACT:=SORTIN
0158 --      1237                                   ELSE ATTR:=SORTIN;
019E --      1238                       SORTIN@.NXTACT:=NEW
01B2 -1      1239                    END
01D0 11      1240                 ELSE BEGIN ATTR:=SORTIN; SORTIN@.NXTACT:=NIL END
0212 -0 B    1241           END; (* OF SORT *)
022C --      1242
022C -- B   1243           PROCEDURE SORTALL(LINK: @ACTIVS);
0044 --      1244           (* DECIDES WETHER TO SORT FIELDNAMES ACCORDING TO SWXRFPRR *)
0044 --      1245           VAR FLDS, OLDFLD: @ACTIVS; PTRTOREC: @TYPES; ANYTIME: BOOLEAN;
0000 0- B    1246           BEGIN SORT(LINK);
002C --      1247              (* CHECK FOR RECORDS (TO PRINT THEIR FIELDS) *)
002C --      1248              (*******************************************)
002C --      1249              (* NOTICE: FIELDS MAY BE LISTED MANY TIMES   *)
002C --      1250              (* AS FAR, I AM UNABLE TO SUPPRESS IT        *)
002C --      1251              (*******************************************)
002C 1-      1252              IF LINK@.IDKLASS IN (.TYPID,VARID,FLDID.) THEN BEGIN
0074 --      1253                 PTRTOREC:=LINK@.TYP; ANYTIME:=FALSE;
00AC --      1254                 WHILE PTRTOREC@.TYPKLASS=ARRTYP
00C0 22      1255                    DO BEGIN PTRTOREC:=PTRTOREC@.COMPTYP; ANYTIME:=TRUE END;
010A --      1256                 IF PTRTOREC@.TYPKLASS=RECTYP THEN
012A --      1257                    (* SUPPRESS MULTIPLE LISTING OF RECORDFIELDS *)
012A --      1258                    (* ON FIRST LEVEL ONLY (RATHER AD-HOC)       *)
012A --      1259                    IF ((PTRTOREC@.SWXRFPRR) AND (LINK@.IDKLASS<>TYPID)) OR
0168 --      1260                       (NOT PTRTOREC@.SWXRFPRR AND (LINK@.IDKLASS=TYPID)) OR ANYTIME
01AE 2-      1261                       THEN BEGIN FLDS:=PTRTOREC@.SECTNS;
01EA --      1262                                  WHILE FLDS<>NIL DO
01F6 3-      1263                                     BEGIN NEW(OLDFLD, FLDID);
020E --      1264                                        OLDFLD@.IDNR:=FLDS@.IDNR;
0250 --      1265                                        OLDFLD@.XREF:=FLDS@.XREF;
0296 --      1266                                        OLDFLD@.TYP:=FLDS@.TYP;
02DC --      1267                                        FLDS:=FLDS@.NXTACT; SORTALL(OLDFLD)
032C -3      1268                                     END
0330 -2      1269                            END
0334 -1      1270                 END
0334 -0 B    1271           END;
0358 --      1272
0000 0- A    1273           BEGIN LINK:=STACK;
002E --      1274              WHILE LINK<>NIL DO
003A 1-      1275                 BEGIN IF LINK@.LEVNR<LEVEL THEN GOTO 1;
0064 --      1276                    PTR:=SRCHTAB(.LINK@.IDNR.);
00B6 --      1277                    (*DE-0 IF PTR@.ACTLSTPTR<>LINK THEN 0-BUG*)
00B6 --      1278                    (*DE-0    WRITELN(DEBUG, 0-BUG*)
00B6 --      1279                    (*DE-0        ' *** SEARCHTABEL IN BAD CONDITION ***'); 0-BUG*)
00B6 --      1280                    STACK:=LINK@.NXTACT;
00EE --      1281                    SRCHTAB(.LINK@.IDNR.):=PTR@.NXTONE;
0154 --      1282                    (* NOTICE: FIELDS FROM WITH STATEMENTS WILL BE    *)
0154 --      1283                    (*         ENTERED INTO THE X-REFERENCE RIGHT NOW *)
0154 --      1284                    SORTALL(LINK);
0176 --      1285                    (* CHECK LABELS*)
0176 --      1286                    IF LINK@.IDKLASS=SCALID THEN
0196 2-      1287                       IF HASHNAME(.LINK@.IDNR.)(.1.)='$' THEN BEGIN
01DC --      1288
01DC --      1289                          FOR I:=1 TO 5 DO LABWORD(.I.):=HASHNAME(.LINK@.IDNR.)(.1+I.);
028E 3-      1290                          IF (LABWORD='LABEL') AND (LINK@.VALUU=0) THEN BEGIN
02CC --      1291                             ERRDESC:='UNSPECIFIED LABEL ''1234''                ';
```

```
02D8 --      1292                          FOR I:=1 TO 4 DO ERRDESC(.19+I.)
0310 --      1293                                           :=HASHNAME(.LINK@.IDNR.)(.6+I.);
0394 -2      1294                          ERRMSG(167, ERRDESC) END END;
03AC --      1295                    LINK:=STACK;
03D0 -1      1296                END; (* WHILE *)
03D4 -0 A    1297         1:END; (* OF POP *)
043C --      1298
043C -- A    1299       FUNCTION SEARCHALL(ID: HASHPTR; IDCAT: IDCLASS): @ACTIVS;
0050 --      1300          VAR RETACT: @ACTIVS; NEWX: @XREFS;
0000 0- A    1301          BEGIN
000A --      1302             IF SRCHTAB(.ID.)=NIL
0024 1-      1303             THEN BEGIN
0036 --      1304                     ERRDESC:='IDENTIFIER ''.........'' UNDECLARED      ';
0042 --      1305                     INSERTWORD:=HASHNAME(.ID.);
006A --      1306                     FOR I:=1 TO 10 DO ERRDESC(.12+I.):=INSERTWORD(.I.);
00EA --      1307                     ERRMSG(104,ERRDESC);
0102 2-      1308                     CASE IDCAT OF
010A --      1309                     SCALID:NEW(RETACT,SCALID); TYPID:NEW(RETACT,TYPID);
0140 --      1310                     VARID :NEW(RETACT,VARID);  FLDID:NEW(RETACT,FLDID);
0178 -2      1311                     PROCID:NEW(RETACT,PROCID); FUNCID:NEW(RETACT,FUNCID) END;
01E2 --      1312                     WITH RETACT@ DO
01F8 2-      1313                        BEGIN IDNR:=ID; LEVNR:=LEVEL;
0220 --      1314                           NEW(XREF); XREF@.NXTREF:=NIL;
024A --      1315                           (* ENTER CORRECT REFERENCE *)
024A --      1316                           IF TOKENNR=SEMICSYM
024A --      1317                              THEN XREF@.OCC:=SCNR-1
0270 --      1318                              ELSE XREF@.OCC:=SCNR;
029C 3-      1319                           CASE IDCAT OF
02A4 44      1320                              SCALID: BEGIN STYP:=UNDFZERO; VALUU:=0; END;
02D2 44      1321                              TYPID: BEGIN NEW(TYP,UNDEF); TYP@.UNDFIDNR:=ID END;
031C --      1322                              VARID, FLDID: TYP:=UNDFZERO;
0344 4-      1323                              PROCID, FUNCID: BEGIN ARG:=NIL;
034A --      1324                                                    (* FLAG IT TO BE UNDEFINED *)
034A --      1325                                                    RETTYP:=NIL;
0350 -4      1326                                                    SWFORW:=FALSE; FORWARGS:=NIL END
035C -3      1327                           END (* OF CASE *)
0392 -2      1328                        END; (* OF WITH *)
0392 --      1329                     IF NOT(RETACT@.IDKLASS IN (.PROCID, FUNCID.)) THEN PUSH(RETACT)
03F8 -1      1330                  END
03FC 1-      1331             ELSE BEGIN RETACT:=SRCHTAB(.ID.)@.ACTLSTPTR;
0452 --      1332                     NEW(NEWX); NEWX@.NXTREF:=RETACT@.XREF;
04A8 --      1333                     RETACT@.XREF:=NEWX;
04DA --      1334                     (* ENTER CORRECT X-REFERENCE *)
04DA --      1335                     IF TOKENNR=SEMICSYM
04DA --      1336                        THEN NEWX@.OCC:=SCNR-1
0500 -1      1337                        ELSE NEWX@.OCC:=SCNR END;
052C --      1338             SEARCHALL:=RETACT
052C -0 A    1339          END; (* SEARCHALL *)
05C0 --      1340
05C0 -- A    1341       PROCEDURE COMPARE(TYP1, TYP2: @TYPES);
0048 --      1342       (* COMPARE TYPES AND PRINT ERROR MESSAGE IF NECESSARY *)
0048 --      1343       VAR SWERR: BOOLEAN; WHATTYP: PACKED ARRAY(.1..7.) OF CHAR;
0053 --      1344           SECLST1, SECLST2: @ACTIVS; J: 0..1;
0000 0- A    1345       BEGIN
000A --      1346          SWERR:=FALSE;
0010 --      1347          IF (TYP1@.TYPKLASS<>UNDEF) AND (TYP2@.TYPKLASS<>UNDEF) THEN
0064 1-      1348             BEGIN
0064 --      1349                IF TYP1@.TYPKLASS<>TYP2@.TYPKLASS
```

```
008C --    1350                  THEN SWERR:=TRUE
0098 2-    1351                  ELSE CASE TYP1@.TYPKLASS OF
00C0 --    1352                       SCALTYP: IF ((TYP1@.SCIDNR<>TYP2@.SCIDNR) OR
00FA --    1353                                    (TYP1@.SCLEVNR<>TYP2@.SCLEVNR)) AND
0136 --    1354                                   (TYP1@.SCIDNR<>0) AND (TYP2@.SCIDNR<>0)
0184 --    1355                                   THEN SWERR:=TRUE
018C --    1356                                   ELSE IF (TYP1@.HIGBND<TYP2@.LOWBND) OR
01D2 --    1357                                           (TYP2@.HIGBND<TYP1@.LOWBND)
020C --    1358                                           THEN SWERR:=TRUE;
0220 3-    1359                       ARRTYP: BEGIN
0220 --    1360                                  COMPARE(TYP1@.COMPTYP,TYP2@.COMPTYP);
0288 --    1361                                  IF (TYP1@.INDXTYP@.TYPKLASS<>UNDEF)
02C2 --    1362                                     AND (TYP2@.INDXTYP@.TYPKLASS<>UNDEF)
02FC --    1363                                     THEN SWERR:=
0304 --    1364                                        (TYP1@.INDXTYP@.SCIDNR<>TYP2@.INDXTYP@.SCIDNR)
0366 --    1365                                      OR(TYP1@.INDXTYP@.SCLEVNR<>TYP2@.INDXTYP@.SCLEVNR)
03C8 --    1366                                      OR (TYP1@.INDXTYP@.LOWBND<>TYP2@.INDXTYP@.LOWBND)
042C --    1367                                      OR (TYP1@.INDXTYP@.HIGBND<>TYP2@.INDXTYP@.HIGBND);
04A8 --    1368                                  SWERR:=SWERR OR (TYP1@.SWPACKA<>TYP2@.SWPACKA)
04E2 -3    1369                               END;
0500 3-    1370                       RECTYP: BEGIN
0500 --    1371                                  SECLST1:=TYP1@.SECTNS; SECLST2:=TYP2@.SECTNS;
0564 --    1372                                  WHILE (SECLST1<>NIL) AND (SECLST2<>NIL) DO
0590 4-    1373                                     BEGIN COMPARE(SECLST1@.TYP,SECLST2@.TYP);
05F8 --    1374                                        SECLST1:=SECLST1@.NXTACT;
062A -4    1375                                        SECLST2:=SECLST2@.NXTACT END;
0660 --    1376                                  SWERR:=(TYP1@.SWPACKR<>TYP2@.SWPACKR)OR
069A --    1377                                         (SECLST1<>NIL) AND (SECLST2<>NIL)
06BE -3    1378                               END;
06DC --    1379                       PTRTYP: IF TYP1<>TYP2 THEN
06E8 --    1380                                  IF COMPNEST<MAXCOMPNST
06E8 3-    1381                                  THEN BEGIN COMPNEST:=COMPNEST+1;
0724 --    1382                                          COMPARE(TYP1@.REFERTYP,TYP2@.REFERTYP);
078C --    1383                                          COMPNEST:=COMPNEST-1
078C -3    1384                                       END
07B6 --    1385                                  ELSE ERRMSG(398,
07C2 --    1386                                           'POINTER TYPES CANNOT BE COMPARED FURTHER');
07D0 -2    1387                       END (* OF CASE *)
07FA -1    1388                  END;
07FA --    1389               IF SWERR
07FA 1-    1390                  THEN BEGIN ERRDESC:='TYPE CONFLICT:''1234567'' VERSUS ''1234567''';
0810 2-    1391                          FOR J:=0 TO 1 DO BEGIN
0832 3-    1392                          CASE TYP1@.TYPKLASS OF
084E --    1393                             SCALTYP: WHATTYP:='SCALAR '; ARRTYP: WHATTYP:='ARRAY  ';
0862 --    1394                             RECTYP: WHATTYP:='RECORD '; PTRTYP: WHATTYP:='POINTER'
086C -3    1395                          END; (* CASE *)
08A0 --    1396                          FOR I:=1 TO 7 DO ERRDESC(.15+J*17+I.):=WHATTYP(.I.);
0928 -2    1397                          TYP1:=TYP2 END (* FOR *);
095A --    1398                          ERRMSG(134, ERRDESC);
0972 -1    1399                     END
0972 -0 A  1400            END; (* COMPARE *)
0A18 --    1401
0A18 -- A  1402            FUNCTION COMPOPNDS(TYP1, TYP2: @TYPES; MUSTBE: INTEGER): BOOLEAN;
0054 --    1403            (* COMPARE OPERANDS ACCORDING TO MUSTBE:                    *)
0054 --    1404            (* IF 1 THEN ARITHMETIC OPERATION (+, -, *, /, MOD, DIV) *)
0054 --    1405            (*    2      BOOLEAN OPERATION (AND, OR)                 *)
0054 --    1406            (*    3      TEST FOR EQUALITY (=, <>)                   *)
0054 --    1407            (*    4      NATURAL ORDERING (<, >, <=, >=)             *)
```

```
0054 --      1408      (*    5      TEST FOR SETELEMENT (IN)                    *)
0054 --      1409      VAR SWERR: BOOLEAN;
0000 0- A    1410      BEGIN
000A --      1411         IF NOT NOTSKIP THEN COMPOPNDS:=FALSE
001E 1-      1412         ELSE BEGIN SWERR:=FALSE;
002E --      1413         IF (TYP1@.TYPKLASS<>UNDEF) AND (TYP2@.TYPKLASS<>UNDEF) THEN
0082 2-      1414            CASE MUSTBE OF
008A 3-      1415            1: BEGIN (* TYPES MUST BE INTEGER SUBRANGES *)
008A --      1416                  IF (TYP1@.TYPKLASS<>SCALTYP) OR (TYP2@.TYPKLASS<>SCALTYP)
00D6 --      1417                     THEN SWERR:=TRUE
00DE --      1418                     ELSE IF (TYP1@.SCIDNR<>SPINTEGER) OR (TYP2@.SCIDNR<>SPINTEGER)
0142 --      1419                             OR (TYP1@.SCLEVNR>0) OR (TYP2@.SCLEVNR>0)
0192 --      1420                             THEN SWERR:=TRUE;
01A2 --      1421                  IF SWERR THEN ERRDESC:='TYPE OF OPERAND(S) MUST BE INTEGER      '
01AC -3      1422               END;
01BC --      1423            2: IF (TYP1<>TYPBOOLEAN) OR (TYP1<>TYPBOOLEAN)
01EC 3-      1424               THEN BEGIN
01F4 --      1425                       SWERR:=TRUE;
01FC --      1426                       ERRDESC:='TYPE OF OPERAND(S) MUST BE BOOLEAN      '
01FC -3      1427                  END;
020C --      1428            3: COMPARE(TYP1, TYP2);
025Q --      1429               (* CAN COMPARE ANYTHING OF EQUAL TYPES *)
0250 --      1430               (* THIS IS NOT STANDARD PASCAL          *)
0250 3-      1431            4: BEGIN
0250 --      1432                  IF (TYP1@.TYPKLASS<>SCALTYP) OR (TYP2@.TYPKLASS<>SCALTYP)
029C --      1433                     THEN SWERR:=TRUE
02A4 --      1434                     ELSE IF (TYP1@.SCIDNR<>TYP2@.SCIDNR) OR
02EA --      1435                             (TYP1@.SCLEVNR<>TYP2@.SCLEVNR)
0324 --      1436                             THEN SWERR:=TRUE;
0334 --      1437                  IF SWERR THEN ERRDESC:=
033E --      1438                               'SAME AND SCALAR OPERANDS EXPECTED      '
033E -3      1439               END;
034E --      1440            5: (* NO SETS IMPLEMENTED, IGNORED *)
034E -1      1441            END; IF SWERR THEN ERRMSG(134, ERRDESC); COMPOPNDS:=NOT SWERR END
03C0 -0 A    1442      END; (* COMPOPNDS *)
0454 --      1443
0454 --      1444      (************************************)
0454 --      1445      (* BEGIN OF CODE GENERATING ROUTINES *)
0454 --      1446      (************************************)
0454 --      1447
0454 -- A    1448      PROCEDURE CODINTEG(N: INTEGER; VAR RETCOD: CHARVAR);
0048 --      1449      (* CONVERTS N TO A VALID LE IN RETCOD *)
0048 --      1450      VAR LG, I, J, K: INTEGER; LITINTEG: PACKED ARRAY(.0..10.) OF CHAR;
0000 0- A    1451      BEGIN
000A --      1452         LG:=CONV(N, LITINTEG);
0026 --      1453         IF LITINTEG(.0.)<>' '
0026 --      1454            THEN J:=LG+12
0036 --      1455            ELSE J:=LG;
004E --      1456         (* WARNING: DIRECT ASSIGNMENT OF STRING *)
004E --      1457         (* EXTREME CARE IS REQUIRED THEREBY!!!! *)
004E --      1458         IF RETCOD@.MAXLG<J
0066 1-      1459            THEN BEGIN FREE(RETCOD);
0098 -1      1460                 LOC(RETCOD, J) END;
00AC --      1461         IF LITINTEG(.0.)<>' '
00AC 1-      1462            THEN BEGIN
00BC --      1463                    LIT(RETCOD,12,'$MINUSUNARY 345');
00D6 --      1464                    K:=RETCOD@.WSPOS+11
00EE -1      1465               END
```

```
00FA --      1466           ELSE K:=RETCOD@.WSPOS-1;
0122 --      1467        FOR I:=1 TO LG DO
0130 --      1468            WORKSPACE(.K+I.):=LITINTEG(.I.);
0192 --      1469        RETCOD@.CURLG:=J;
01B2 --      1470        IF LITINTEG(.0.)<>' '
01B2 --      1471           THEN WRAP('(',RETCOD,')')
01DA -0 A    1472     END; (* CODINTEG *)
0218 --      1473
0218 -- A    1474     PROCEDURE WRAPNRS (* VAR PIECE: CHARVAR *);
0044 --      1475     (* PUTS A PAIR OF PARENTHESIS AND A NOT ENCLOSED NUMBER *)
0044 --      1476     (* ATTACHED AROUND PIECE. THIS ROUTINE MAKES THE CODE   *)
0044 --      1477     (* MORE READABLE. THE NUMBERS ARE IN SUPERSCRIPT FORM.  *)
0044 --      1478        CONST ZEROSUP=176; (* ORDER OF SUPERSCRIPT 0 *)
0044 --      1479        VAR NR, NRFND: INTEGER;
004C --      1480            POS, STEND: 1..MAXWSL;
0054 --      1481            RESULT, NRLIT: CHARVAR;
0000 0- A    1482     BEGIN
000A --      1483        LOC(RESULT, PIECE@.CURLG+6);
003A --      1484        LOC(NRLIT, 2);
004E --      1485        NR := 0;
0054 --      1486        (* SCAN PIECE FOR ALL SUPERSCRIPTS INSIDE *)
0054 --      1487        POS:=PIECE@.WSPOS;
0086 --      1488        STEND:=POS+PIECE@.CURLG-1;
00C0 --      1489        WHILE POS<=STEND DO
00CC --      1490           IF (ORD(WORKSPACE(.POS.))<ZEROSUP)
00FE --      1491                 OR (ORD(WORKSPACE(.POS.))>ZEROSUP+9)
0136 --      1492              THEN POS:=POS+1
013E 1-      1493              ELSE BEGIN
0160 --      1494                   NRFND:=ORD(WORKSPACE(.POS.))-ZEROSUP;
018C --      1495                   POS:=POS+1;
01AA --      1496                   WHILE (ORD(WORKSPACE(.POS.))>=ZEROSUP)
01DC --      1497                            AND (ORD(WORKSPACE(.POS.))<=ZEROSUP+9)
0214 --      1498                            AND (POS<=STEND) DO
0230 2-      1499                      BEGIN
0230 --      1500                         NRFND:=10*NRFND+ORD(WORKSPACE(.POS.))-ZEROSUP;
0266 --      1501                         POS:=POS+1
0266 -2      1502                      END;
0288 --      1503                   IF NR<=NRFND THEN NR:=NRFND+1;
02A0 --      1504                   POS:=POS+1 (* POS WAS TESTED ALREADY *)
02A0 -1      1505                   END;
02C2 --      1506        CODINTEG(NR, NRLIT);
02D6 --      1507        (* CONVERT TO SUPERSCRIPT *)
02D6 --      1508        FOR I:=1 TO NRLIT@.CURLG DO
02F8 --      1509            WORKSPACE(.NRLIT@.WSPOS+I-1.):=CHR( ORD(WORKSPACE(.NRLIT@.WSPOS+I-1.))
037A --      1510                                                +ZEROSUP-ORD('0') );
03B4 --      1511        (* CONSTRUCT RESULT *)
03B4 --      1512        LIT(RESULT, 1, '(23456789012345');
03CE --      1513        APP(RESULT, NRLIT);
03F8 --      1514        APP(RESULT, PIECE);
0426 --      1515        APP(RESULT, NRLIT);
0450 --      1516        (* MISUSE PIECE *) LIT(PIECE, 1, ')23456789012345');
046A --      1517        APP(RESULT, PIECE);
0498 --      1518        (* THROW OLD PIECE AWAY *) FREE(PIECE);
04BE --      1519        (* MAKE IT RESULT *) PIECE:=RESULT
04BE -0 A    1520     END; (* WRAPNRS *)
0528 --      1521
0528 -- A    1522     PROCEDURE CODIDENT(PTRACT: @ACTIVS; VAR RETCOD: CHARVAR);
0048 --      1523     (* ASSUMES THAT PTRACT REFERS TO A VARIABLE, FUNCTION, OR PROCEDURE *)
```

```
0048 --      1524      (* IDENTIFIER. RETURNS ITS NAME AND EV. A LEVEL NUMBER ATTACHED TO  *)
0048 --      1525      (* IT IN CASE THERE IS A NAME CONFLICT.                              *)
0048 --      1526      VAR ID: HASHPTR; COUNT: INTEGER; SCANACTLST: @ACTLST;
0054 --      1527          SUBSCRIPT: CHARVAR;
0000 0- A    1528      BEGIN
000A --      1529         ID:=PTRACT@.IDNR;
0038 --      1530         (* WARNING: DIRECT ASSIGNMENT TO STRING *)
0038 --      1531         (* EXERCISE EXTREME CARE THEREBY        *)
0038 --      1532         IF RETCOD@.MAXLG<10
0050 11      1533            THEN BEGIN FREE(RETCOD); LOC(RETCOD, 10) END;
0096 --      1534         COUNT:=0;
009C --      1535         WHILE (COUNT<10) AND (HASHNAME(.ID.)(.COUNT MOD 10 + 1.)<>' ') DO
0118 1-      1536            BEGIN COUNT:=COUNT+1;
0124 --      1537               WORKSPACE(.RETCOD@.WSPOS+COUNT-1.):=HASHNAME(.ID.)(.COUNT.)
018E -1      1538            END;
01AA --      1539         RETCOD@.CURLG:=COUNT;
01CA --      1540         (* SEARCH FOR A CONFLICT *)
01CA --      1541         SCANACTLST:=SRCHTAB(.ID.);
0208 --      1542         COUNT:=0;
020E --      1543         WHILE SCANACTLST<>NIL DO
021A 1-      1544            BEGIN
021A --      1545               IF SCANACTLST@.ACTLSTPTR@.IDKLASS IN (.VARID, FUNCID, PROCID.)
0242 --      1546                  THEN COUNT:=COUNT+1;
0282 --      1547               SCANACTLST:=SCANACTLST@.NXTONE
0296 -1      1548            END;
02B8 --      1549         (* ATTACH LEVEL NUMBER IN THIS CASE *)
02B8 --      1550         (* EXCEPTION: LEVEL 0 (I.E. "INPUT", "OUTPUT", "CHR" ETC. *)
02B8 --      1551         IF (COUNT>1) AND (PTRACT@.LEVNR<>0) THEN
02F8 1-      1552            BEGIN
02F8 --      1553               APP(RETCOD, DOLLAR); LOC(SUBSCRIPT, 3);
033C --      1554               CODINTEG(PTRACT@.LEVNR, SUBSCRIPT);
0364 --      1555               APP(RETCOD, SUBSCRIPT); FREE(SUBSCRIPT)
03AC -1      1556            END
03B0 -0 A    1557      END; (* CODIDENT *)
03EC --      1558
03EC -- A    1559      PROCEDURE ENVIRON(VAR RETCOD: CHARVAR);
0044 --      1560      (* THIS PROCEDURE RETURNS THE CURRENT ENVIRONMENT OF VARIABLES.    *)
0044 --      1561      (* THE FORMAT IS "@ $PHI,V1,V2,...:" WHERE "@" STANDS FOR LAMBDA  *)
0044 --      1562      (* AND VI'S STAND FOR ALL VARIABLE NAMES (NAME CONFLICTS REMOVED).*)
0044 --      1563      (* FUNCTION RETURN VARIABLES ARE THEIR FUNCTION IDENTIFIER AP-    *)
0044 --      1564      (* PENDED TO "$PI$" (IN CONTRARY TO DEFINITIONS AND CALLS).       *)
0044 --      1565      (* THIS PROCEDURE DOES NOT INCLUDE ADDITIONAL NAMES NEEDED FOR    *)
0044 --      1566      (* SIDE EFFECT TRANSLATION. SORRY!                                *)
0044 --      1567      VAR STACKTRAV: @ACTIVS; CURNAME: CHARVAR;
0000 0- A    1568      BEGIN
000A --      1569         LOC(CURNAME, 20);
001E --      1570         LIT(RETCOD,6,'@ $PHI789012345');
0038 --      1571         STACKTRAV:=STACK;
005C --      1572         WHILE STACKTRAV<>NIL DO
0068 1-      1573            BEGIN
0068 2-      1574               CASE STACKTRAV@.IDKLASS OF
0084 --      1575               SCALID, TYPID, PROCID:;(* IGNORE *)
0088 3-      1576               VARID, FLDID: BEGIN
0088 --      1577                                CODIDENT(STACKTRAV, CURNAME);
00B2 --      1578                                APP(RETCOD, LITCOMMA);
00E2 --      1579                                APP(RETCOD, CURNAME)
0108 -3      1580                             END;
0110 --      1581               FUNCID:IF (STACKTRAV@.LEVNR<>0) AND
```

```
0136 --      1582                        (STACKTRAV@.LEVNR<LEVEL) THEN
016A 3-      1583                        BEGIN
016A --      1584                           CODIDENT(STACKTRAV, CURNAME);
0194 --      1585                           APP(RETCOD, LITCOMMA);
01C4 --      1586                           APP(RETCOD, PI);
01F4 --      1587                           APP(RETCOD, CURNAME)
021A -3      1588                        END
021E -2      1589                  END; (* OF CASE *)
0254 --      1590                  STACKTRAV:=STACKTRAV@.NXTACT
0268 -1      1591               END;
028A --      1592            APP(RETCOD, LITCOLON);
02BA --      1593            FREE(CURNAME)
02D8 -0 A    1594         END; (* ENVIRON *)
0300 --      1595
0300 -- A    1596         PROCEDURE APPENV(VAR COD: CHARVAR);
0044 --      1597         (* APPENDS A LIST OF THE CURRENT ENVIRONMENT TO COD *)
0044 --      1598         VAR STACKTRAV: @ACTIVS; APPENDIX: CHARVAR;
0000 0- A    1599         BEGIN
000A --      1600            LOC(APPENDIX, 10);
001E --      1601            LIT(APPENDIX, 4, '$PHI56789012345');
0038 --      1602            APP(COD, APPENDIX);
0062 --      1603            IF COD@.CURLG>4 THEN WRAP('(',COD,')')
009E --      1604                           ELSE APP(COD, LITBLANK);
00D6 --      1605            STACKTRAV:=STACK;
00FA --      1606            WHILE STACKTRAV<>NIL DO
0106 1-      1607               BEGIN
0106 2-      1608                  CASE STACKTRAV@.IDKLASS OF
0122 --      1609                  SCALID, TYPID, PROCID:;
0126 --      1610                  VARID, FLDID:
0126 3-      1611                     BEGIN
0126 --      1612                        CODIDENT(STACKTRAV, APPENDIX);
0150 --      1613                        APP(COD, APPENDIX);
017A --      1614                        WRAP('(',COD,')')
0192 -3      1615                     END;
019A --      1616                  FUNCID:
019A --      1617                     IF (STACKTRAV@.LEVNR<>0) AND
01C0 --      1618                        (STACKTRAV@.LEVNR<LEVEL)
01EC 3-      1619                        THEN BEGIN
01F4 --      1620                             CODIDENT(STACKTRAV, APPENDIX);
021E --      1621                             CAT(APPENDIX, PI, APPENDIX);
026C --      1622                             APP(COD, APPENDIX);
0296 --      1623                             WRAP('(',COD,')');
02B2 -3      1624                             END
02B2 -2      1625                  END; (*CASE*)
02E8 --      1626                  STACKTRAV:=STACKTRAV@.NXTACT
02FC -1      1627               END;
031E --      1628            FREE(APPENDIX);
0340 -0 A    1629         END; (*APPENV*)
0368 --      1630
0368 -- A    1631         PROCEDURE STMOUT(NR: INTEGER; STMCOD: CHARVAR);
0048 --      1632         (* PREFIXES STMCOD WITH "$STM<STMNR><EQUIV>", APPENDS <TERMIN> *)
0048 --      1633         (* AND PUTS THE COMPLETE CODE ONTO SPUNCH                      *)
0048 --      1634         VAR PREFIX: CHARVAR;
0000 0- A    1635         BEGIN
000A --      1636            LOC(PREFIX, 8);
001E --      1637            CODINTEG(NR, PREFIX);
0032 --      1638            CAT(PREFIX, STM, PREFIX);
0080 --      1639            APP(PREFIX, EQUIV);
```

```
00B0 --      1640          APP(STMCOD, TERMIN);
00E0 --      1641          CAT(STMCOD, PREFIX, STMCOD);
0128 --      1642          OUT(STMCOD)
0146 -0 A    1643       END; (* STMOUT *)
0154 --      1644
0154 --      1645       (************************************)
0154 --      1646       (* PRODUCTIONS OF GRAMMAR START HERE *)
0154 --      1647       (************************************)
0154 --      1648
0154 -- A    1649       PROCEDURE TYPEPROC(VAR RETTYP: @TYPES; REFID: HASHPTR); FORWARD;
0048 --      1650
0048 -- A    1651       PROCEDURE FIELDLIST(VAR FLDLST: @ACTIVS); FORWARD;
0044 --      1652
0044 -- A    1653       PROCEDURE EXPRESSION(VAR EXPCOD: CHARVAR; VAR EXPTYP: @TYPES); FORWARD;
0048 --      1654
0048 -- A    1655       PROCEDURE STATEMENT; FORWARD;
0040 --      1656
0040 -- A    1657       PROCEDURE BLOCK; FORWARD;
0040 --      1658
0040 -- A    1659       PROCEDURE IDENTLIST(FIRSTID: HASHPTR; VAR CARIDLST: @IDLST);
0048 --      1660       (* <IDENTIFIER_LIST> *)
0048 --      1661       VAR CDRIDLST: @IDLST; CURIDNR: INTEGER;
0000 0- A    1662       BEGIN
000A --      1669          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1670          IF NOTSKIP THEN
001A --      1671          IF TOKENNR IN (.COMMASYM,RPARASYM,SEMICSYM,COLONSYM.)
001A 1-      1672             THEN BEGIN
0058 2-      1673                     CASE TOKENNR OF
0066 --      1674                        COMMASYM:
0066 3-      1675                           BEGIN
0066 --      1676                              ADVANCE;
006A --      1677                              TERMINAL(IDENTIFIER,CURIDNR);
007E --      1678                              IDENTLIST(CURIDNR,CDRIDLST)
00A0 -3      1679                           END;
00A8 --      1680                        RPARASYM,SEMICSYM,COLONSYM:
00A8 --      1681                           (* EPSILON PRODUCTION *)
00A8 --      1682                           CDRIDLST:=NIL
00A8 -2      1683                     END;(* OF CASE *)
00F4 --      1684                     NEW(CARIDLST); CARIDLST@.ID:=FIRSTID;
013A --      1685                     CARIDLST@.NXTID:=CDRIDLST
0152 -1      1686                END
0170 1-      1687             ELSE BEGIN
0174 --      1688                     ERRMSG26(TOKENNR,'IDENTIFIERLIST   ');
019E --      1689                     SYNCHRONIZE; NOTSKIP:=FALSE; CARIDLST:=NIL
01AE -1      1690                END
01B8 --      1691       ELSE CARIDLST:=NIL (* RETURN NIL IN ERRORMODE *)
01BC -0 A    1692       END; (* OF IDENTLIST *)
020C --      1693
020C -- A    1694       PROCEDURE LABELDCL;
0040 --      1695       (* <LABEL_DECLARATION> *)
0040 --      1696       VAR LABVAL: INTEGER; LABNR: HASHPTR;
0048 --      1697
0048 -- B    1698       PROCEDURE LABLDCLREM;
0040 --      1699       (* <LABEL_DECLARATION_REMAINDER> *)
0000 0- B    1700       BEGIN
000A --      1707          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1708          IF TOKENNR IN (.COMMASYM,SEMICSYM.)
000A 1-      1709             THEN BEGIN
```

```
0048 2-      1710                    CASE TOKENNR OF
0056 --      1711                       COMMASYM:
0056 3-      1712                          BEGIN
0056 --      1713                             ADVANCE;
005A --      1714                             TERMINAL(UNSGINTEG,LABVAL);
006E --      1715                             LABNR:=CONVPSID(LABVAL,'LABEL');
009A --      1716                             IF SEARCHONLY(LABNR) THEN PUSH(BLDLAB(LABNR));
0102 --      1717                             LABLDCLREM
0102 -3      1718                          END;
010A --      1719                       SEMICSYM:
010A 3-      1720                          BEGIN
010A --      1721                             NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0118 --      1722                             ADVANCE
0118 -3      1723                          END
011C -2      1724                    END (* OF CASE *)
0142 -1      1725                 END
0142 1-      1726              ELSE BEGIN
0146 --      1727                    ERRMSG26(TOKENNR,'LABELDCLREMAINDER');
0170 --      1728                    GOTO 9999
017E -1      1729                 END
017E -0 B    1730        END; (* OF LABLDCLREM *)
01C8 --      1731
0000 0- A    1732        BEGIN
000A --      1733        (* <LABEL_DECLARATION> *)
000A --      1740           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1741           IF NOTSKIP THEN
001A --      1742           IF TOKENNR IN (.LABELSYM,CONSTSYM,TYPESYM,VARSYM,PROCSYM,FUNCSYM,
001A --      1743                          BEGINSYM.)
001A 1-      1744              THEN BEGIN
0054 2-      1745                    CASE TOKENNR OF
0062 --      1746                       LABELSYM:
0062 3-      1747                          BEGIN
0062 --      1748                             ADVANCE;
0066 --      1749                             TERMINAL(UNSGINTEG,LABVAL);
007A --      1750                             LABNR:=CONVPSID(LABVAL,'LABEL');
00A6 --      1751                             IF SEARCHONLY(LABNR) THEN PUSH(BLDLAB(LABNR));
010E --      1752                             LABLDCLREM
010E -3      1753                          END;
0116 --      1754                       CONSTSYM,TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM:
0116 --      1755                          (* EPSILON PRODUCTION *)
0116 -2      1756                    END (* OF CASE *)
01B0 -1      1757                 END
01B0 1-      1758              ELSE BEGIN
01B4 --      1759                    ERRMSG26(TOKENNR,'LABELDECLARATION ');
01DE --      1760                    SYNCHRONIZE; NOTSKIP:=FALSE
01E2 -1      1761                 END
01EE -0 A    1762        END; (* OF LABELDCL *)
0234 --      1763
0234 -- A    1764        PROCEDURE NONIDCONST(VAR CONSTYP: @TYPES; VAR CONSVAL: INTEGER);
0048 --      1765        (* <NON_IDENTIFIER_CONSTANT> *)
0048 --      1766        VAR PTRACT: @ACTIVS; MSIGN: -1..1;
0050 --      1767
0050 -- B    1768        PROCEDURE NONIDCOREM;
0040 --      1769        (* <NON_IDENTIFIER_CONSTANT_REMAINDER> *)
0000 0- B    1770        BEGIN
000A --      1777           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1778           IF NOTSKIP THEN
001A --      1779           IF TOKENNR IN (.IDENTIFIER,UNSGINTEG,UNSGREAL.)
```

```
001A 1-      1780            THEN BEGIN
0058 2-      1781                    CASE TOKENNR OF
0066 3-      1782                       IDENTIFIER: BEGIN
0066 --      1783                          PTRACT:=SEARCHALL(TOKENFD,SCALID);
00AE --      1784                          IF PTRACT@.IDKLASS=SCALID
00C2 4-      1785                             THEN BEGIN CONSTYP:=PTRACT@.STYP;
0104 -4      1786                                    CONSVAL:=PTRACT@.VALUU END
0124 4-      1787                             ELSE BEGIN ERRMSG103(TOKENFD);
014C --      1788                                    NEW(CONSTYP,UNDEF);
0168 --      1789                                    CONSTYP@.UNDFIDNR:=SPCONST;
01A0 -4      1790                                    CONSVAL:=0 END;
01AA -3      1791                          ADVANCE END;
01B2 3-      1792                       UNSGINTEG: BEGIN CONSTYP:=TYPINTEGER;
01DA --      1793                          CONSVAL:=TOKENFD;
01EC -3      1794                          ADVANCE END;
01F4 3-      1795                       UNSGREAL: BEGIN NEW(CONSTYP,UNDEF);
0210 --      1796                          CONSTYP@.UNDFIDNR:=SPREAL; CONSVAL:=TOKENFD;
025A -3      1797                          ADVANCE END
025E -2      1798                    END (* OF CASE *)
02D0 -1      1799                END
02D0 1-      1800            ELSE BEGIN
02D4 --      1801                    ERRMSG26(TOKENNR,'NONIDENTCONSTREM ');
02FE --      1802                    SYNCHRONIZE; NOTSKIP:=FALSE; NEW(CONSTYP,UNDEF);
032A --      1803                    CONSTYP@.UNDFIDNR:=SPCONST; CONSVAL:=0
0362 -1      1804                END
036C -0 B    1805        END; (* OF NONIDCOREM *)
03B4 --      1806
0000 0- A    1807        BEGIN
000A --      1808        (* <NON_IDENTIFIER_CONSTANT> *)
000A --      1815           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1816           IF NOTSKIP THEN
001A --      1817           IF TOKENNR IN (.PLUSMINUS,UNSGINTEG,UNSGREAL,STRINGSYM.)
001A 1-      1818              THEN BEGIN
0058 2-      1819                    CASE TOKENNR OF
0066 --      1820                       PLUSMINUS:
0066 3-      1821                          BEGIN MSIGN:=(-2)*TOKENFD+3;
008E --      1822                             ADVANCE;
0092 --      1823                             NONIDCOREM;
0096 --      1824                             IF CONSTYP@.TYPKLASS=ARRTYP
00AE --      1825                                THEN ERRMSG(105,
00C2 --      1826                                    'SIGN NOT ALLOWED                    ')
00C8 --      1827                                ELSE CONSVAL:=MSIGN*CONSVAL
00D0 -3      1828                          END;
00E8 3-      1829                       UNSGINTEG: BEGIN CONSTYP:=TYPINTEGER;
0110 --      1830                          CONSVAL:=TOKENFD;
0122 -3      1831                          ADVANCE END;
012A 3-      1832                       UNSGREAL: BEGIN NEW(CONSTYP, UNDEF);
0146 --      1833                          CONSTYP@.UNDFIDNR:=SPREAL; CONSVAL:=TOKENFD;
0190 -3      1834                          ADVANCE END;
0198 3-      1835                       STRINGSYM: BEGIN
0198 --      1836                          IF TOKENFD MOD MAXSTRGL=1
01AA 4-      1837                             THEN BEGIN
01B2 --      1838                                    CONSTYP:=TYPCHAR;
01DA --      1839                                    CONSVAL:=ORD(STRING(.TOKENFD DIV MAXSTRGL.));
020E -4      1840                                 END
020E 4-      1841                             ELSE BEGIN
0212 --      1842                                    CONSTYP:=BLDSTRG(TOKENFD);
0246 --      1843                                    CONSVAL:=TOKENFD DIV MAXSTRGL
```

```
0246 -4      1844                                END;
0260 -3      1845                             ADVANCE END
0264 -2      1846                          END (* OF CASE *)
0292 -1      1847                       END
0292 1-      1848                 ELSE BEGIN
0296 --      1849                       ERRMSG26(TOKENNR,'NONIDENTCONSTANT ');
02C0 --      1850                       SYNCHRONIZE; NOTSKIP:=FALSE; NEW(CONSTYP, UNDEF);
02EC --      1851                       CONSTYP@.UNDFIDNR:=SPCONST; CONSVAL:=0
0324 -1      1852                    END
032E -0 A    1853          END; (* OF NONIDCONST *)
03B4 --      1854
03B4 -- A    1855          PROCEDURE CONSTANT(VAR CONSTYP: @TYPES; VAR CONSVAL: INTEGER);
0048 --      1856          (* <CONSTANT> *)
0048 --      1857          VAR PTRACT: @ACTIVS;
0000 0- A    1858          BEGIN
000A --      1865             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1866             IF NOTSKIP THEN
001A --      1867             IF TOKENNR IN (.IDENTIFIER,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM.)
001A 1-      1868                THEN BEGIN
0058 2-      1869                        CASE TOKENNR OF
0066 3-      1870                           IDENTIFIER: BEGIN
0066 --      1871                              PTRACT:=SEARCHALL(TOKENFD, SCALID);
00AE --      1872                              IF PTRACT@.IDKLASS=SCALID
00C2 4-      1873                                 THEN BEGIN CONSTYP:=PTRACT@.STYP;
0104 -4      1874                                         CONSVAL:=PTRACT@.VALUU END
0124 4-      1875                                 ELSE BEGIN ERRMSG103(TOKENFD);
014C --      1876                                         NEW(CONSTYP, UNDEF);
0168 --      1877                                         CONSTYP@.UNDFIDNR:=SPCONST;
01A0 -4      1878                                         CONSVAL:=0 END;
01AA -3      1879                              ADVANCE END;
01B2 --      1880                           UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM:
01B2 --      1881                              NONIDCONST(CONSTYP, CONSVAL)
01C2 -2      1882                        END (* OF CASE *)
023C -1      1883                     END
023C 1-      1884                ELSE BEGIN
0240 --      1885                        ERRMSG26(TOKENNR,'CONSTANT         ');
026A --      1886                        SYNCHRONIZE; NOTSKIP:=FALSE; NEW(CONSTYP,UNDEF);
0296 --      1887                        CONSTYP@.UNDFIDNR:=SPCONST; CONSVAL:=0
02CE -1      1888                     END
02D8 -0 A    1889          END; (* OF CONSTANT *)
0320 --      1890
0320 -- A    1891          PROCEDURE CONSTLIST(FIRSTTYP: @TYPES; FIRSTVAL: INTEGER;
0040 --      1892                              VAR CARCONSLST: @VALUS; MATCHTYP: @TYPES);
0050 --      1893          (* <CONSTANT_LIST> *)
0050 --      1894          VAR CONSTYP: @TYPES; CONSVAL:INTEGER; CDRCONSLST: @VALUS; SW: BOOLEAN;
0000 0- A    1895          BEGIN
000A --      1902             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1903             IF NOTSKIP THEN
001A --      1904             IF TOKENNR IN (.COMMASYM,COLONSYM.)
001A 1-      1905                THEN BEGIN
0058 2-      1906                        CASE TOKENNR OF
0066 --      1907                           COMMASYM:
0066 3-      1908                              BEGIN
0066 --      1909                                 ADVANCE;
006A --      1910                                 CONSTANT(CONSTYP,CONSVAL);
007E --      1911                                 CONSTLIST(CONSTYP,CONSVAL,CDRCONSLST,MATCHTYP);
00CE -3      1912                              END;
00D2 --      1913                           COLONSYM:
```

```
00D2 --      1914                          (* EPSILON PRODUCTION *)
00D2 --      1915                          CDRCONSLST:=NIL
00D2 -2      1916                     END;(* OF CASE *)
010E --      1917                     IF (FIRSTTYP@.TYPKLASS=SCALTYP) AND
0134 --      1918                        (MATCHTYP@.TYPKLASS=SCALTYP)
015A --      1919                        THEN SW:=(FIRSTTYP@.SCIDNR<>MATCHTYP@.SCIDNR) OR
019C --      1920                                 (FIRSTTYP@.SCLEVNR<>MATCHTYP@.SCLEVNR)
01D6 --      1921                        ELSE SW:=TRUE; (* IF STRING AS CASE LABEL OR UNDEFINED *)
01FA --      1922                        IF SW
01FA 2-      1923                           THEN BEGIN IF (MATCHTYP@.TYPKLASS<>UNDEF) AND
022A --      1924                                         (FIRSTTYP@.TYPKLASS<>UNDEF)
0250 --      1925                                         THEN ERRMSG(147,
0260 --      1926                                           'CASE LABEL TYPE DOES NOT MATCH SELECTOR ');
026A -2      1927                                 CARCONSLST:=CDRCONSLST END
028C 2-      1928                           ELSE BEGIN NEW(CARCONSLST);
02A4 --      1929                                CARCONSLIST@.VALEE:=FIRSTVAL;
02C4 -2      1930                                CARCONSLST@.NXTVAL:=CDRCONSLST END
02FA -1      1931                     END
02FA 1-      1932                ELSE BEGIN
02FE --      1933                        ERRMSG26(TOKENNR,'CONSTANTLIST     ');
0328 --      1934                        SYNCHRONIZE; NOTSKIP:=FALSE; CARCONSLST:=NIL
0338 -1      1935                     END
0342 --      1936             ELSE CARCONSLST:=NIL (* RETURN NIL IN ERRORMODE, TOO *)
0346 -0 A    1937          END; (* OF CONSTLIST *)
03C0 --      1938
03C0 -- A    1939          PROCEDURE CONSDEFPRT;
0040 --      1940          (* <CONSTANT_DEFINITION_PART> *)
0040 --      1941          VAR ID:HASHPTR; CONSTYP: @TYPES; CONSVAL: INTEGER; PTRACT: @ACTIVS;
0050 --      1942
0050 -- B    1943          PROCEDURE CONSDFPTRM;
0040 --      1944          (* <CONSTANT_DEFINITION_PART_REMAINDER> *)
0000 0- B    1945          BEGIN
000A --      1952             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1953             IF NOTSKIP THEN
001A --      1954             IF TOKENNR IN (.IDENTIFIER,TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-      1955                THEN BEGIN
0054 2-      1956                        CASE TOKENNR OF
0062 --      1957                           IDENTIFIER:
0062 3-      1958                              BEGIN NEW(PTRACT,SCALID);
0078 --      1959                                 WITH PTRACT@ DO
008E 4-      1960                                    BEGIN IDNR:=TOKENFD; LEVNR:=LEVEL;
00BC --      1961                                       IDKLASS:=SCALID; NEW(XREF);
00D2 -4      1962                                       XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL END;
0108 --      1963                                 ADVANCE;
010C --      1964                                 TERMINAL(EQUALSYM,DUMMYFD);
0122 --      1965                                 IF NOT NOTSKIP THEN
0136 4-      1966                                    BEGIN NEW(CONSTYP,UNDEF);
014E -4      1967                                       CONSTYP@.UNDFIDNR:=SPCONST; CONSVAL:=0 END;
0188 --      1968                                 CONSTANT(CONSTYP,CONSVAL);
019C --      1969                                 PTRACT@.STYP:=CONSTYP; PTRACT@.VALUU:=CONSVAL;
01EA --      1970                                 IF SEARCHONLY(PTRACT@.IDNR) THEN PUSH(PTRACT);
0248 --      1971                                 TERMINAL(SEMICSYM,DUMMYFD);
025E --      1972                                 CONSDFPTRM
025E -3      1973                              END;
0266 --      1974                           TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM:
0266 --      1975                              (* EPSILON PRODUCTION *)
0266 -2      1976                        END (* OF CASE *)
031C -1      1977                     END
```

```
031C 1-      1978            ELSE BEGIN
0320 --      1979                    ERRMSG26(TOKENNR,'CONSTDEFINPARTREM');
034A --      1980                    SYNCHRONIZE; NOTSKIP:=FALSE
034E -1      1981                END
035A -0 B    1982      END; (* OF CONSDFPTRM *)
03AC --      1983
0000 0- A    1984      BEGIN
000A --      1985      (* <CONSTANT_DEFINITION_PART> *)
000A --      1992         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      1993         IF NOTSKIP THEN
001A --      1994         IF TOKENNR IN (.CONSTSYM,TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-      1995            THEN BEGIN
0054 2-      1996                    CASE TOKENNR OF
0062 --      1997                       CONSTSYM:
0062 3-      1998                          BEGIN
0062 --      1999                             ADVANCE;
0066 --      2000                             TERMINAL(IDENTIFIER,ID);
007A --      2001                             NEW(PTRACT,SCALID);
0090 --      2002                             WITH PTRACT@ DO
00A6 4-      2003                                BEGIN IDNR:=ID; LEVNR:=LEVEL;
00CE --      2004                                   IDKLASS:=SCALID; NEW(XREF);
00E4 -4      2005                                   XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL END;
011A --      2006                             TERMINAL(EQUALSYM,DUMMYFD);
0130 --      2007                             IF NOT NOTSKIP THEN
0144 4-      2008                                BEGIN NEW(CONSTYP,UNDEF);
015C -4      2009                                   CONSTYP@.UNDFIDNR:=SPCONST; CONSVAL:=0 END;
0196 --      2010                             CONSTANT(CONSTYP, CONSVAL);
01AA --      2011                             PTRACT@.STYP:=CONSTYP; PTRACT@.VALUU:=CONSVAL;
01F8 --      2012                             IF SEARCHONLY(PTRACT@.IDNR) THEN PUSH(PTRACT);
0256 --      2013                             TERMINAL(SEMICSYM,DUMMYFD);
026C --      2014                             CONSDFPTRM
026C -3      2015                          END;
0274 --      2016                       TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM:
0274 --      2017                          (* EPSILON PRODUCTION *)
0274 -2      2018                    END (* OF CASE *)
030E -1      2019                END
030E 1-      2020             ELSE BEGIN
0312 --      2021                    ERRMSG26(TOKENNR,'CONSTANTDEFINPART');
033C --      2022                    SYNCHRONIZE; NOTSKIP:=FALSE
0340 -1      2023                END
034C -0 A    2024      END; (* OF CONSDEFPRT *)
039C --      2025
039C -- A    2026      PROCEDURE SIMPLETYPE(VAR RETTYP: @TYPES; REFID: HASHPTR);
0048 --      2027      (* <SIMPLE_TYPE> *)
0048 --      2028      VAR PTRACT: @ACTIVS; SCIDLST: @IDLST; ID: HASHPTR;
0054 --      2029          CONSTYP1, CONSTYP2: @TYPES; CONSVAL1, CONSVAL2: INTEGER;
0064 --      2030          ORDER, ORDCNT: INTEGER; NXTSCID: @IDLST;
0070 --      2031
0070 -- B    2032      PROCEDURE SUBRANGE;
0000 0- B    2033      BEGIN
000A 1-      2034         IF NOTSKIP THEN BEGIN
001A --      2035            IF (CONSTYP1@.TYPKLASS<>SCALTYP) OR
0040 --      2036               (CONSTYP2@.TYPKLASS<>SCALTYP) THEN
006E 2-      2037               BEGIN IF (CONSTYP1@.TYPKLASS<>UNDEF) AND
0094 --      2038                        (CONSTYP2@.TYPKLASS<>UNDEF) THEN ERRMSG(148,
00CA --      2039                           'SUBRANGE BOUNDS MUST BE SCALAR          ');
00D4 --      2040                  NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=REFID
0108 -2      2041               END ELSE
```

```
0126 --      2042            IF (CONSTYP1@.SCIDNR<>CONSTYP2@.SCIDNR) OR
0160 --      2043               (CONSTYP1@.SCLEVNR<>CONSTYP2@.SCLEVNR) THEN
01A2 2-      2044               BEGIN ERRMSG(145,
01AA --      2045                             'TYPE CONFLICT                              ');
01B4 --      2046                  NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=REFID
01E8 -2      2047               END ELSE
0206 --      2048            IF CONSVAL1>CONSVAL2 THEN
0212 2-      2049               BEGIN ERRMSG(102,
021A --      2050                             'LOWBOUND EXCEEDS HIGHBOUND                 ');
0224 --      2051                  NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=REFID
0258 -2      2052               END ELSE
0276 2-      2053               BEGIN NEW(RETTYP,SCALTYP);
0290 --      2054                  RETTYP@.SCIDNR:=CONSTYP1@.SCIDNR;
02D6 --      2055                  RETTYP@.SCLEVNR:=CONSTYP1@.SCLEVNR;
030A --      2056                  RETTYP@.LOWBND:=CONSVAL1; RETTYP@.HIGBND:=CONSVAL2
0342 -2      2057               END
034A -1      2058            END
034A --      2059            ELSE (* NOT NOTSKIP *)
034E 1-      2060                 BEGIN NEW(RETTYP, UNDEF);
036A -1      2061                    RETTYP@.UNDFIDNR:=REFID END
039C -0 B    2062      END; (* SUBRANGE *)
0430 --      2063
0430 -- B    2064      PROCEDURE SIMPTYPREM(FIRSTID: HASHPTR);
0044 --      2065      (* <SIMPLE_TYPE_REMAINDER> *)
0000 0- B    2066      BEGIN
000A --      2073         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2074         IF NOTSKIP THEN
001A --      2075         IF TOKENNR IN (.DOUBLEDOT,RPARASYM,SEMICSYM,COMMASYM,RBRACKSYM,
001A --      2076                        ENDSYM.)
001A 1-      2077            THEN BEGIN
0054 2-      2078                    CASE TOKENNR OF
0062 --      2079                       DOUBLEDOT:
0062 3-      2080                          BEGIN
0062 --      2081                             PTRACT:=SEARCHALL(FIRSTID, SCALID);
00A4 --      2082                             ADVANCE;
00A8 --      2083                             CONSTANT(CONSTYP2, CONSVAL2);
00BC --      2084                             IF NOTSKIP
00BC 4-      2085                                THEN BEGIN
00CC --      2086                                        IF PTRACT@.IDKLASS<>SCALID
00E0 5-      2087                                           THEN BEGIN ERRMSG103(FIRSTID);
010A --      2088                                                      NEW(RETTYP, UNDEF);
0126 -5      2089                                                      RETTYP@.UNDFIDNR:=REFID END
0158 5-      2090                                           ELSE BEGIN CONSTYP1:=PTRACT@.STYP;
018E --      2091                                                      CONSVAL1:=PTRACT@.VALUU;
01AA -5      2092                                                      SUBRANGE END
01AE -4      2093                                     END
01AE 4-      2094                                ELSE BEGIN NEW(RETTYP, UNDEF);
01CE -4      2095                                           RETTYP@.UNDFIDNR:=REFID END
0200 -3      2096                          END;
0204 --      2097                       RPARASYM,SEMICSYM,COMMASYM,RBRACKSYM,ENDSYM:
0204 --      2098                          (* EPSILON PRODUCTION *)
0204 3-      2099                          BEGIN PTRACT:=SEARCHALL(FIRSTID, TYPID);
0248 --      2100                             IF PTRACT@.IDKLASS<>TYPID
025C 4-      2101                                THEN BEGIN ERRMSG103(FIRSTID);
0286 --      2102                                           NEW(RETTYP, UNDEF);
02A2 -4      2103                                           RETTYP@.UNDFIDNR:=REFID END
02D4 --      2104                                ELSE RETTYP:=PTRACT@.TYP
02EC -3      2105                          END
```

```
030E -2      2106                    END (* OF CASE *)
03E4 -1      2107                 END
03E4 1-      2108              ELSE BEGIN
03E8 --      2109                    ERRMSG26(TOKENNR,'SIMPLETYPEREMAIND');
0412 --      2110                    SYNCHRONIZE; NOTSKIP:=FALSE
0416 -1      2111                 END
0422 -0 B    2112        END; (* OF SIMPTYPREM *)
0468 --      2113
0000 0- A    2114        BEGIN
000A --      2115        (* <SIMPLE_TYPE> *)
000A --      2122           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2123           IF NOTSKIP THEN
001A --      2124           IF TOKENNR IN (.LPARASYM,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM,
001A --      2125                          IDENTIFIER.)
001A 1-      2126              THEN BEGIN
0058 2-      2127                      CASE TOKENNR OF
0066 --      2128                         LPARASYM:
0066 3-      2129                            BEGIN
0066 --      2130                               ADVANCE;
006A --      2131                               TERMINAL(IDENTIFIER,ID);
007E --      2132                               IDENTLIST(ID,SCIDLST);
00A4 --      2133                               (* SCALAR ORDERS START FROM 0 *)
00A4 --      2134                               ORDER:=-1; ORDCNT:=-1; NXTSCID:=SCIDLST;
00D2 --      2135                               (* COUNT SCALAR IDENTIFIERS FIRST *)
00D2 --      2136                               WHILE NXTSCID<>NIL DO
00DE 4-      2137                                   BEGIN ORDCNT:=ORDCNT+1;NXTSCID:=NXTSCID@.NXTID
00FE -4      2138                                   END;
0120 --      2139                               WHILE SCIDLST<>NIL DO
012C 4-      2140                                  BEGIN ORDER:=ORDER+1;
0138 --      2141                                     IF SEARCHONLY(SCIDLST@.ID) THEN
0174 5-      2142                                        BEGIN NEW(PTRACT, SCALID);
018A --      2143                                           NEW(PTRACT@.STYP, SCALTYP);
01B4 --      2144                                           WITH PTRACT@, STYP@ DO
01E0 6-      2145                                              BEGIN IDNR:=SCIDLST@.ID;
020E --      2146                                                 LEVNR:=LEVEL; VALUU:=ORDER;
0224 --      2147                                                 NEW(XREF); XREF@.OCC:=SCNR;
0250 --      2148                                                 XREF@.NXTREF:=NIL; VALUU:=ORDER;
0272 --      2149                                                 IF REFID=0 THEN
027E 7-      2150                                                    BEGIN IMPLTYPNR:=IMPLTYPNR+1;
0296 --      2151                                                       REFID:=CONVPSID(IMPLTYPNR,
02A4 --      2152                                                                       'IMPLT')
02AA -7      2153                                                    END;
02C8 --      2154                                                 SCIDNR:=REFID; SCLEVNR:=LEVEL;
02F0 --      2155                                                 LOWBND:=0; HIGBND:=ORDCNT;
02FE --      2156                                                 PUSH(PTRACT)
031C -6      2157                                              END(* OF WITH *)
0320 -5      2158                                        END; (* OF IF *)
0320 --      2159                                     SCIDLST:=SCIDLST@.NXTID
0334 -4      2160                                  END; (* OF WHILE *)
0356 --      2161                               NEW(RETTYP,SCALTYP);
0370 --      2162                               RETTYP@.SCIDNR:=REFID; RETTYP@.SCLEVNR:=LEVEL;
03C8 --      2163                               RETTYP@.LOWBND:=0; RETTYP@.HIGBND:=ORDCNT;
0406 --      2164                               TERMINAL(RPARASYM,DUMMYFD)
0418 -3      2165                            END;
0420 --      2166                         UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM:
0420 3-      2167                            BEGIN
0420 --      2168                               NONIDCONST(CONSTYP1,CONSVAL1);
0434 --      2169                               TERMINAL(DOUBLEDOT,DUMMYFD);
```

```
044A --    2170                              CONSTANT(CONSTYP2,CONSVAL2);
045E --    2171                              SUBRANGE
045E -3    2172                           END;
0466 --    2173                        IDENTIFIER:
0466 3-    2174                           BEGIN ID:=TOKENFD;
0486 --    2175                              ADVANCE;
048A --    2176                              SIMPTYPREM(ID)
04A4 -3    2177                           END
04A8 -2    2178                     END (* OF CASE *)
051E -1    2179                  END
051E 1-    2180               ELSE BEGIN
0522 --    2181                     ERRMSG26(TOKENNR,'SIMPLETYPE       ');
054C --    2182                     SYNCHRONIZE; NOTSKIP:=FALSE;
055C --    2183                     NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=REFID
0590 -1    2184                  END
05AA -0 A  2185         END; (* OF SIMPLETYPE *)
060C --    2186
060C -- A  2187         PROCEDURE SIMPTYPLST(FIRSTTYP: @TYPES; VAR CARSMPLST: @TYPES; SWPACK: BOOLEAN);
004C --    2188         (* <SIMPLE_TYPE_LIST> *)
004C --    2189         VAR CDRSMPLST: @TYPES; CURTYP: @TYPES;
0000 0- A  2190         BEGIN
000A --    2197            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    2198            IF NOTSKIP THEN
001A --    2199            IF TOKENNR IN (.COMMASYM,RBRACKSYM.)
001A 1-    2200               THEN BEGIN
0058 2-    2201                     CASE TOKENNR OF
0066 --    2202                        COMMASYM:
0066 3-    2203                           BEGIN
0066 --    2204                              ADVANCE;
006A --    2205                              SIMPLETYPE(CURTYP,0);
007C --    2206                              SIMPTYPLST(CURTYP,CDRSMPLST,SWPACK)
00BC -3    2207                           END;
00C4 --    2208                        RBRACKSYM:
00C4 --    2209                           (* EPSILON PRODUCTION *)
00C4 --    2210                           CDRSMPLST:=NIL
00C4 -2    2211                     END;(* OF CASE *)
00F4 --    2212                     NEW(CARSMPLST, ARRTYP); CARSMPLST@.SWPACKA:=SWPACK;
0142 --    2213                     IF FIRSTTYP@.TYPKLASS<>SCALTYP THEN
0162 --    2214                        IF FIRSTTYP@.TYPKLASS<>UNDEF
0176 2-    2215                           THEN BEGIN
0182 --    2216                                   ERRMSG(113,
018A --    2217                                    'INDEX TYPE MUST BE SCALAR OR SUBRANGE   ');
0194 --    2218                                   CARSMPLST@.INDXTYP:=UNDFZERO
01AC -2    2219                              END
01D0 --    2220                           ELSE CARSMPLST@.INDXTYP:=FIRSTTYP
01EC --    2221                        ELSE (* FIRSTTYP IS SCALAR *)
020E --    2222                     IF FIRSTTYP=TYPINTEGER
020E 2-    2223                        THEN BEGIN ERRMSG(149,
0228 --    2224                                           'INDEX TYPE MUST NOT BE INTEGER          ');
0232 --    2225                                 CARSMPLST@.INDXTYP:=UNDFZERO
024A -2    2226                              END
026E --    2227                        ELSE CARSMPLST@.INDXTYP:=FIRSTTYP;
02A8 --    2228                     CARSMPLST@.COMPTYP:=CDRSMPLST
02C0 -1    2229                  END
02DE 1-    2230               ELSE BEGIN
02E2 --    2231                     ERRMSG26(TOKENNR,'SIMPLETYPELIST   ');
030C --    2232                     SYNCHRONIZE; NOTSKIP:=FALSE; CARSMPLST:=NIL
031C -1    2233                  END
```

```
0326 --      2234          ELSE CARSMPLST:=NIL (* RETURN NIL IN ERRORMODE, TOO *)
032A -0 A    2235       END; (* OF SIMPTYPLST *)
03D0 --      2236
03D0 -- A    2237       PROCEDURE VARIANT(VAR RECLST: @ACTIVS; MATCHTYP: @TYPES);
0048 --      2238       (* <VARIANT> *)
0048 --      2239       VAR CONSTYP: @TYPES; CONSVAL: INTEGER; CONSLST: @VALUS;
0000 0- A    2240       BEGIN
000A --      2247          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2248          IF NOTSKIP THEN
001A --      2249          IF TOKENNR IN (.IDENTIFIER,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM,
001A --      2250                         RPARASYM,SEMICSYM,ENDSYM.)
001A 1-      2251             THEN BEGIN
0054 2-      2252                     CASE TOKENNR OF
0062 --      2253                        IDENTIFIER,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM:
0062 3-      2254                           BEGIN
0062 --      2255                              CONSTANT(CONSTYP,CONSVAL);
0076 --      2256                              CONSTLIST(CONSTYP,CONSVAL,CONSLST,MATCHTYP);
00C6 --      2257                              TERMINAL(COLONSYM,DUMMYFD);
00DC --      2258                              TERMINAL(LPARASYM,DUMMYFD);
00F2 --      2259                              FIELDLIST(RECLST);
00FE --      2260                              TERMINAL(RPARASYM,DUMMYFD)
0110 -3      2261                           END;
0118 --      2262                        RPARASYM,SEMICSYM,ENDSYM:
0118 --      2263                           (* EPSILON PRODUCTION *)
0118 --      2264                           RECLST:=NIL
0118 -2      2265                     END (* OF CASE *)
0210 -1      2266                END
0210 1-      2267             ELSE BEGIN
0214 --      2268                     ERRMSG26(TOKENNR,'VARIANT          ');
023E --      2269                     SYNCHRONIZE; NOTSKIP:=FALSE; RECLST:=NIL
024E -1      2270                END
0258 -0 A    2271       END; (* OF VARIANT *)
0294 --      2272
0294 -- A    2273       PROCEDURE VARIANTLST(FIRSTVAR:@ACTIVS;VAR CARRECLST:@ACTIVS;MATCHTYP:@TYPES);
004C --      2274       (* <VARIANT_LIST> *)
004C --      2275       VAR CURVAR, OLD, NEW, CDRRECLST: @ACTIVS;
0000 0- A    2276       BEGIN
000A --      2283          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2284          IF TOKENNR IN (.SEMICSYM,RPARASYM,ENDSYM.)
000A 1-      2285             THEN BEGIN
0044 2-      2286                     CASE TOKENNR OF
0052 --      2287                        SEMICSYM:
0052 3-      2288                           BEGIN
0052 --      2289                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0060 --      2290                              ADVANCE;
0064 --      2291                              VARIANT(CURVAR, MATCHTYP);
008E --      2292                              VARIANTLST(CURVAR, CDRRECLST, MATCHTYP);
00D6 -3      2293                           END;
00DA --      2294                        RPARASYM,ENDSYM:
00DA 3-      2295                           BEGIN
00DA --      2296                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
00E8 --      2297                              (* EPSILON PRODUCTION *)
00E8 --      2298                              CDRRECLST:=NIL
00E8 -3      2299                           END
00EE -2      2300                     END;(* OF CASE *)
01B0 --      2301                     OLD:=NIL; NEW:=FIRSTVAR;
01D4 --      2302                     WHILE NEW<>NIL DO
01E0 22      2303                        BEGIN OLD:=NEW; NEW:=NEW@.NXTACT END;
```

```
0234 --    2304                    IF OLD<>NIL
0234 22    2305                       THEN BEGIN OLD@.NXTACT:=CDRRECLST; CARRECLST:=FIRSTVAR END
0294 --    2306                       ELSE CARRECLST:=CDRRECLST
0298 -1    2307                 END
02BA 1-    2308              ELSE BEGIN
02BE --    2309                       ERRMSG26(TOKENNR,'VARIANTLIST      ');
02E8 --    2310                       GOTO 9999
02F6 -1    2311                 END
02F6 -0 A  2312      END; (* OF VARIANTLST *)
0330 --    2313
0330 -- A  2314      PROCEDURE TAGFIELDRM(FIRSTID: HASHPTR; VAR REC: @ACTIVS; VAR MATCHTYP: @TYPES);
004C --    2315      (* <TAG_FIELD_REMAINDER> *)
004C --    2316      VAR PTRACT: @ACTIVS; MATCHID: HASHPTR;
0000 0- A  2317      BEGIN
000A --    2324         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    2325         IF NOTSKIP THEN
001A --    2326         IF TOKENNR IN (.COLONSYM,OFSYM.)
001A 1-    2327            THEN BEGIN
0054 2-    2328                    CASE TOKENNR OF
0062 --    2329                       COLONSYM:
0062 3-    2330                          BEGIN
0062 --    2331                             NEW(REC, FLDID);
007E --    2332                             REC@.IDNR:=FIRSTID; REC@.LEVNR:=0;
00CE --    2333                             NEW(REC@.XREF); REC@.XREF@.OCC:=SCNR;
012A --    2334                             REC@.XREF@.NXTREF:=NIL;
015C --    2335                             ADVANCE;
0160 --    2336                             TERMINAL(IDENTIFIER,MATCHID)
0170 -3    2337                          END;
0178 --    2338                       OFSYM:
0178 --    2339                          (* EPSILON PRODUCTION *)
0178 33    2340                          BEGIN REC:=NIL; MATCHID:=FIRSTID END
019C -2    2341                    END;(* OF CASE *)
024A --    2342                    IF MATCHID<>0 THEN
0256 2-    2343                       BEGIN
0256 --    2344                       PTRACT:=SEARCHALL(MATCHID, TYPID);
029A --    2345                       IF PTRACT@.IDKLASS<>TYPID
02AE 33    2346                          THEN BEGIN ERRMSG103(MATCHID); MATCHID:=0 END
02DE --    2347                          ELSE IF NOT(PTRACT@.TYP@.TYPKLASS IN (.SCALTYP,UNDEF.))
0344 3-    2348                                  THEN BEGIN ERRMSG(110,
0346 --    2349                                             'TAGFIELD TYPE MUST BE SCALAR OR SUBRANGE');
0350 -3    2350                                             MATCHID:=0 END
0356 -2    2351                       END;
0356 --    2352                    IF MATCHID<>0
0356 --    2353                       THEN MATCHTYP:=PTRACT@.TYP
0376 --    2354                       ELSE MATCHTYP:=UNDFZERO;
03C4 --    2355                    IF REC<>NIL THEN
03D4 --    2356                       REC@.TYP:=MATCHTYP
03EC -1    2357                 END
040E 1-    2358              ELSE BEGIN
0412 --    2359                       ERRMSG26(TOKENNR,'TAGFIELDREMAINDER');
043C --    2360                       SYNCHRONIZE; NOTSKIP:=FALSE; REC:=NIL; MATCHTYP:=UNDFZERO
0456 -1    2361                 END
047E -0 A  2362      END; (* OF TAGFIELDRM *)
04FC --    2363
04FC -- A  2364      PROCEDURE FIELDLSTRM(VAR RECLST: @ACTIVS);
0044 --    2365      (* <FIELD_LIST_REMAINDER> *)
0000 0- A  2366      BEGIN
000A --    2373         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
```

```
000A --  2374          IF TOKENNR IN (.SEMICSYM,RPARASYM,ENDSYM.)
000A 1-  2375             THEN BEGIN
0044 2-  2376                     CASE TOKENNR OF
0052 --  2377                        SEMICSYM:
0052 3-  2378                           BEGIN
0052 --  2379                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0060 --  2380                              ADVANCE;
0064 --  2381                              FIELDLIST(RECLST)
006C -3  2382                           END;
0074 --  2383                        RPARASYM,ENDSYM:
0074 --  2384                           (* EPSILON PRODUCTION *)
0074 3-  2385                           BEGIN
0074 --  2386                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0082 --  2387                              RECLST:=NIL
0082 -3  2388                           END
008C -2  2389                     END (* OF CASE *)
014E -1  2390                END
014E 1-  2391             ELSE BEGIN
0152 --  2392                     ERRMSG26(TOKENNR,'FIELDLISTREMAIND ');
017C --  2393                     GOTO 9999
018A -1  2394                END
018A -0 A 2395       END; (* OF FIELDLSTRM *)
01C4 --  2396
01C4 -- A 2397       PROCEDURE RECORDSECT(VAR RECLST: @ACTIVS);
0044 --  2398       (* <RECORD_SECTION> *)
0044 --  2399       VAR FIRSTID: HASHPTR; FLDIDLST: @IDLST; FLDTYP: @TYPES; LASTACT: @ACTIVS;
0000 0- A 2400       BEGIN
000A --  2407          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --  2408          IF NOTSKIP THEN
001A --  2409          IF TOKENNR IN (.IDENTIFIER,RPARASYM,SEMICSYM,ENDSYM.)
001A 1-  2410             THEN BEGIN
0054 2-  2411                     CASE TOKENNR OF
0062 --  2412                        IDENTIFIER:
0062 3-  2413                           BEGIN FIRSTID:=TOKENFD;
0082 --  2414                              ADVANCE;
0086 --  2415                              IDENTLIST(FIRSTID,FLDIDLST);
00AC --  2416                              TERMINAL(COLONSYM,DUMMYFD);
00C2 --  2417                              (* GET UNDEF-TYPE IF WE GOT SOME ERROR SO FAR *)
00C2 --  2418                              IF NOT NOTSKIP THEN FLDTYP:=UNDFZERO;
00FA --  2419                              TYPEPROC(FLDTYP,0); RECLST:=NIL;
0116 --  2420                              WHILE FLDIDLST<>NIL DO
0122 4-  2421                                 BEGIN LASTACT:=RECLST; NEW(RECLST, FLDID);
0160 --  2422                                    WITH RECLST@ DO
017A 5-  2423                                       BEGIN IDNR:=FLDIDLST@.ID; LEVNR:=0;
01AE --  2424                                          NEW(XREF); XREF@.OCC:=SCNR;
01DA --  2425                                          XREF@.NXTREF:=NIL; TYP:=FLDTYP
01F4 -5  2426                                       END;
0212 --  2427                                    RECLST@.NXTACT:=LASTACT;
0248 --  2428                                    FLDIDLST:=FLDIDLST@.NXTID
025C -4  2429                                 END
027A -3  2430                           END;
0282 --  2431                        RPARASYM,SEMICSYM,ENDSYM:
0282 --  2432                           (* EPSILON PRODUCTION *)
0282 --  2433                           RECLST:=NIL
0282 -2  2434                     END (* OF CASE *)
034E -1  2435                END
034E 1-  2436             ELSE BEGIN
0352 --  2437                     ERRMSG26(TOKENNR,'RECORDSECTION    ');
```

```
037C --    2438                    SYNCHRONIZE; NOTSKIP:=FALSE; RECLST:=NIL
038C -1    2439                END
0396 -0 A  2440        END; (* OF RECORDSECT *)
03E8 --    2441
03E8 -- A  2442        PROCEDURE FIELDLIST;
0044 --    2443        (* <FIELD_LIST> *)
0044 --    2444        VAR CDRRECLST, CURVAR, OLD, NEW: @ACTIVS;
0054 --    2445            FIRSTID: HASHPTR; MATCHTYP: @TYPES;
0000 0- A  2446        BEGIN
000A --    2453           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    2454           IF TOKENNR IN (.IDENTIFIER,RPARASYM,SEMICSYM,ENDSYM,CASESYM.)
000A 1-    2455              THEN BEGIN
0044 2-    2456                      CASE TOKENNR OF
0052 --    2457                         IDENTIFIER,RPARASYM,SEMICSYM,ENDSYM:
0052 3-    2458                            BEGIN
0052 --    2459                               NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0060 --    2460                               RECORDSECT(FLDLST);
006C --    2461                               FIELDLSTRM(CDRRECLST);
0078 --    2462                               OLD:=NIL; NEW:=FLDLST;
00A0 44    2463                               WHILE NEW<>NIL DO BEGIN OLD:=NEW; NEW:=NEW@.NXTACT END;
0100 --    2464                               IF OLD<>NIL
0100 --    2465                                  THEN OLD@.NXTACT:=CDRRECLST
0120 --    2466                                  ELSE FLDLST:=CDRRECLST
0142 -3    2467                            END;
0168 --    2468                         CASESYM:
0168 3-    2469                            BEGIN
0168 --    2470                               ERRMSG(398,'VARIANTS WILL BE TREATED AS RECORDS     ');
017A --    2471                               ADVANCE;
017E --    2472                               TERMINAL(IDENTIFIER,FIRSTID);
0192 --    2473                               TAGFIELDRM(FIRSTID,FLDLST,MATCHTYP);
01C0 --    2474                               TERMINAL(OFSYM,DUMMYFD);
01D6 --    2475                               VARIANT(CURVAR,MATCHTYP);
0200 --    2476                               VARIANTLST(CURVAR,CDRRECLST,MATCHTYP);
0248 --    2477                               IF FLDLST<>NIL
0248 --    2478                                  THEN FLDLST@.NXTACT:=CDRRECLST
0270 --    2479                                  ELSE FLDLST:=CDRRECLST
0292 -3    2480                            END
02B4 -2    2481                      END (* OF CASE *)
038E -1    2482                END
038E 1-    2483             ELSE BEGIN
0392 --    2484                      ERRMSG26(TOKENNR,'FIELDLIST        ');
03BC --    2485                      GOTO 9999
03CA -1    2486                END
03CA -0 A  2487        END; (* OF FIELDLIST *)
0434 --    2488
0434 -- A  2489        PROCEDURE UNPKSTRTYP(VAR RETTYP: @TYPES; SWPACK: BOOLEAN; REFID: HASHPTR);
004C --    2490        (* <UNPACKED_STRUCTURED_TYPE> *)
004C --    2491        VAR INDXTYPLST, LASTCOMP, COMPONENT, CURTYP: @TYPES; FLDLST: @ACTIVS;
0060 --    2492            RECEL1, RECEL2: @ACTIVS;
0068 --    2493            (*DE-3 DEBUGREC: @ACTIVS; 3-BUG*)
0000 0- A  2494        BEGIN
000A --    2501           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    2502           IF NOTSKIP THEN
001A --    2503           IF TOKENNR IN (.ARRAYSYM,RECORDSYM,FILESYM,SETSYM.)
001A 1-    2504              THEN BEGIN
0054 2-    2505                      CASE TOKENNR OF
0062 --    2506                         ARRAYSYM:
0062 3-    2507                            BEGIN
```

```
0062 --    2508                          ADVANCE;
0066 --    2509                          TERMINAL(LBRACKSYM,DUMMYFD);
007C --    2510                          SIMPLETYPE(CURTYP,0);
008E --    2511                          SIMPTYPLST(CURTYP,INDXTYPLST,SWPACK);
00D2 --    2512                          TERMINAL(RBRACKSYM,DUMMYFD);
00E8 --    2513                          TERMINAL(OFSYM,DUMMYFD);
00FE --    2514                          IF NOT NOTSKIP THEN COMPONENT:=UNDFZERO;
0136 --    2515                          TYPEPROC(COMPONENT,0);
0148 --    2516                          LASTCOMP:=NIL; RETTYP:=INDXTYPLST;
0170 --    2517                          WHILE INDXTYPLST<>NIL DO
017C 4-    2518                             BEGIN LASTCOMP:=INDXTYPLST;
019A -4    2519                                INDXTYPLST:=INDXTYPLST@.COMPTYP END;
01D0 --    2520                          IF LASTCOMP=NIL
01D0 4-    2521                             THEN BEGIN NEW(RETTYP, ARRTYP);
01F8 --    2522                                     RETTYP@.SWPACKA:=SWPACK;
022A --    2523                                     RETTYP@.INDXTYP:=UNDFZERO;
0266 -4    2524                                     LASTCOMP:=RETTYP END;
0288 --    2525                          LASTCOMP@.COMPTYP:=COMPONENT
029C -3    2526                       END;
02BE --    2527                    RECORDSYM:
02BE 3-    2528                       BEGIN NEW(RETTYP, RECTYP); RETTYP@.SWPACKR:=SWPACK;
030C --    2529                          RETTYP@.SWXRFPRR:=(REFID=0); NEST:=NEST+1;
0364 --    2530                          ADVANCE;
0368 --    2531                          FIELDLIST(FLDLST); RETTYP@.SECTNS:=FLDLST;
03AA --    2532                          (* CHECK FOR SAME FIELDS *)
03AA --    2533                          RECEL1:=FLDLST;
03C8 4-    2534                          WHILE RECEL1<>NIL DO BEGIN
03D4 --    2535                             (* FOR NOW LET IT BE A VARIABLE *)
03D4 --    2536                             RECEL1@.PARM:=VARB;
03EE --    2537                             RECEL2:=RECEL1@.NXTACT;
0420 5-    2538                             WHILE RECEL2<>NIL DO BEGIN
042C 6-    2539                                IF RECEL1@.IDNR=RECEL2@.IDNR THEN BEGIN
0460 --    2540                             ERRDESC:='TWO RECORDFIELDS ''1234567890''            ';
046C --    2541                             INSERTWORD:=HASHNAME(.RECEL1@.IDNR.);
04A8 --    2542                             FOR I:=1 TO 10 DO ERRDESC(.18+I.):=INSERTWORD(.I.);
0528 -6    2543                             ERRMSG(101, ERRDESC) END;
0540 -5    2544                                RECEL2:=RECEL2@.NXTACT END;
0576 -4    2545                             RECEL1:=RECEL1@.NXTACT END;
05AC --    2559                          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
05AC --    2560                          TERMINAL(ENDSYM,DUMMYFD); NEST:=NEST-1;
05DA -3    2561                       END;
05DE --    2562                    FILESYM:
05DE 3-    2563                       BEGIN ERRMSG(398,
05E6 --    2564                                'FILES ARE NOT SUPPORTED                ');
05F0 --    2565                          ADVANCE;
05F4 --    2566                          TERMINAL(OFSYM,DUMMYFD);
060A --    2567                          TYPEPROC(RETTYP,0);
061C --    2568                          NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=SPFILE;
0670 -3    2569                       END;
0674 --    2570                    SETSYM:
0674 3-    2571                       BEGIN ERRMSG(398,
067C --    2572                                'SETS ARE NOT SUPPORTED                 ');
0686 --    2573                          ADVANCE;
068A --    2574                          TERMINAL(OFSYM,DUMMYFD);
06A0 --    2575                          SIMPLETYPE(RETTYP,0);
06B2 --    2576                          NEW(RETTYP, UNDEF); RETTYP@.UNDFIDNR:=SPSET;
0706 -3    2577                       END
0706 -2    2578                 END (* OF CASE *)
```

```
0790 -1      2579                END
0790 1-      2580             ELSE BEGIN
0794 --      2581                ERRMSG26(TOKENNR,'UNPACKSTRUCTTYPE ');
07BE --      2582                SYNCHRONIZE; NOTSKIP:=FALSE; RETTYP:=UNDFZERO
07CE -1      2583                END
07F6 -0 A    2584       END; (* OF UNPKSTRTYP *)
08D4 --      2585
08D4 -- A    2586       PROCEDURE TYPEPROC; (* (VAR RETTYP: @TYPES; REFID: HASHPTR) *)
0048 --      2587       (* <TYPE> *)
0048 --      2588       VAR ID: HASHPTR; FORWELEM: @FORWLST; PTRACT: @ACTIVS;
0000 0- A    2589       BEGIN
000A --      2596          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2597          IF NOTSKIP THEN
001A --      2598          IF TOKENNR IN (.IDENTIFIER,LPARASYM,UNSGINTEG,PLUSMINUS,UNSGREAL,
001A --      2599                         STRINGSYM,PACKEDSYM,ARRAYSYM,RECORDSYM,FILESYM,
001A --      2600                         SETSYM,POINTER.)
001A 1-      2601             THEN BEGIN
0054 2-      2602                CASE TOKENNR OF
0062 --      2603                   IDENTIFIER,LPARASYM,UNSGINTEG,PLUSMINUS,UNSGREAL,
0062 --      2604                   STRINGSYM:
0062 --      2605                      SIMPLETYPE(RETTYP,REFID);
008C --      2606                   PACKEDSYM:
008C 3-      2607                      BEGIN
008C --      2608                         ADVANCE;
0090 --      2609                         UNPKSTRTYP(RETTYP,TRUE,REFID)
00BA -3      2610                      END;
00C2 --      2611                   ARRAYSYM,RECORDSYM,FILESYM,SETSYM:
00C2 --      2612                      UNPKSTRTYP(RETTYP,FALSE,REFID);
00F2 --      2613                   POINTER:
00F2 3-      2614                      BEGIN NEW(RETTYP,PTRTYP);
010E --      2615                         ADVANCE;
0112 --      2616                         TERMINAL(IDENTIFIER,ID); RETTYP@.PTRIDNR:=ID;
0158 --      2617                         IF NOT NOTSKIP THEN RETTYP@.REFERTYP:=UNDFZERO
0184 4-      2618                         ELSE BEGIN
01AC 5-      2619                         IF ALLOWFW THEN BEGIN
01BC --      2620                         (* EVERY REFERENCE IS TREATED FORWARD *)
01BC --      2621                         NEW(FORWELEM); FORWELEM@.LOCTYP:=RETTYP;
0202 --      2622                         (* ENTER CORRECT X-REFERENCE *)
0202 --      2623                         IF TOKENNR=SEMICSYM
0202 --      2624                            THEN FORWELEM@.LOCSCNR:=SCNR-1
0228 --      2625                            ELSE FORWELEM@.LOCSCNR:=SCNR;
0254 -5      2626                         FORWELEM@.NEXTF:=FORWTYPS; FORWTYPS:=FORWELEM END
02B0 5-      2627                         ELSE BEGIN
02B4 --      2628                         PTRACT:=SEARCHALL(ID, TYPID);
02F8 --      2629                         IF PTRACT@.IDKLASS=TYPID
030C --      2630                            THEN RETTYP@.REFERTYP:=PTRACT@.TYP
0344 6-      2631                            ELSE BEGIN RETTYP@.REFERTYP:=UNDFZERO;
03A2 -5      2632                                    ERRMSG103(ID) END END
03C0 -4      2633                         END (* ELSE *)
03C0 -3      2634                      END
03C0 -2      2635                END (* OF CASE *)
04CE -1      2636             END
04CE 1-      2637             ELSE BEGIN
04D2 --      2638                ERRMSG26(TOKENNR,'TYPE             ');
04FC --      2639                SYNCHRONIZE; NOTSKIP:=FALSE; NEW(RETTYP, UNDEF);
0528 --      2640                RETTYP@.UNDFIDNR:=REFID
0540 -1      2641             END
055A -0 A    2642       END; (* OF TYPEPROC *)
```

```
05AC --       2643
05AC -- A     2644      PROCEDURE TYPEDEFPRT;
0040 --       2645      (* <TYPE_DEFINITION_PART> *)
0040 --       2646      VAR ID: HASHPTR; TYPTYP: @TYPES; PTRACT: @ACTIVS;
004C --       2647
004C -- B     2648      PROCEDURE TYPEDFPTRM;
0040 --       2649      (* <TYPE_DEFINITION_PART_REMAINDER> *)
0000 0- B     2650      BEGIN
000A --       2657         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --       2658         IF NOTSKIP THEN
001A --       2659         IF TOKENNR IN (.IDENTIFIER,VARSYM,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-       2660            THEN BEGIN
0054 2-       2661                    CASE TOKENNR OF
0062 --       2662                       IDENTIFIER:
0062 3-       2663                          BEGIN
0062 --       2664                             NEW(PTRACT, TYPID);
007A --       2665                             WITH PTRACT@ DO
0090 4-       2666                                BEGIN IDNR:=TOKENFD; LEVNR:=LEVEL;
00BE --       2667                                   NEW(XREF); XREF@.OCC:=SCNR;
00EA -4       2668                                   XREF@.NXTREF:=NIL END;
0104 --       2669                             ADVANCE;
0108 --       2670                             TERMINAL(EQUALSYM,DUMMYFD);
011E --       2671                             IF NOT NOTSKIP THEN
0132 4-       2672                                BEGIN NEW(TYPTYP, UNDEF);
014A -4       2673                                   TYPTYP@.UNDFIDNR:=PTRACT@.IDNR END;
018C --       2674                             TYPEPROC(TYPTYP,PTRACT@.IDNR); PTRACT@.TYP:=TYPTYP;
01F8 --       2675                             IF SEARCHONLY(PTRACT@.IDNR) THEN PUSH(PTRACT);
0256 --       2676                             TERMINAL(SEMICSYM,DUMMYFD);
026C --       2677                             TYPEDFPTRM
026C -3       2678                          END;
0274 --       2679                       VARSYM,PROCSYM,FUNCSYM,BEGINSYM:
0274 --       2680                          (* EPSILON PRODUCTION *)
0274 -2       2681                    END (* OF CASE *)
032A -1       2682                END
032A 1-       2683            ELSE BEGIN
032E --       2684                    ERRMSG26(TOKENNR,'TYPEDEFINPARTREM ');
0358 --       2685                    SYNCHRONIZE; NOTSKIP:=FALSE
035C -1       2686                END
0368 -0 B     2687      END; (* OF TYPEDFPTRM *)
03B8 --       2688
0000 0- A     2689      BEGIN
000A --       2690      (* <TYPE_DEFINITION_PART> *)
000A --       2697         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --       2698         IF NOTSKIP THEN
001A --       2699         IF TOKENNR IN (.TYPESYM,VARSYM,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-       2700            THEN BEGIN
0054 2-       2701                    CASE TOKENNR OF
0062 --       2702                       TYPESYM:
0062 3-       2703                          BEGIN
0062 --       2704                             ADVANCE;
0066 --       2705                             TERMINAL(IDENTIFIER,ID);
007A --       2706                             NEW(PTRACT, TYPID);
0092 --       2707                             WITH PTRACT@ DO
00A8 4-       2708                                BEGIN IDNR:=ID; LEVNR:=LEVEL;
00D0 --       2709                                   NEW(XREF); XREF@.OCC:=SCNR;
00FC -4       2710                                   XREF@.NXTREF:=NIL END;
0116 --       2711                             TERMINAL(EQUALSYM,DUMMYFD);
012C 4-       2712                             IF NOT NOTSKIP THEN BEGIN
```

```
0140 -4      2713                              NEW(TYPTYP, UNDEF); TYPTYP@.UNDFIDNR:=ID END;
0186 --      2714                            TYPEPROC(TYPTYP,ID); PTRACT@.TYP:=TYPTYP;
01DE --      2715                            IF SEARCHONLY(ID) THEN PUSH(PTRACT);
0228 --      2716                            TERMINAL(SEMICSYM,DUMMYFD);
023E --      2717                            TYPEDFPTRM
023E -3      2718                         END;
0246 --      2719                      VARSYM,PROCSYM,FUNCSYM,BEGINSYM:
0246 --      2720                         (* EPSILON PRODUCTION *)
0246 -2      2721                   END (* OF CASE *)
02E0 -1      2722                END
02E0 1-      2723             ELSE BEGIN
02E4 --      2724                   ERRMSG26(TOKENNR,'TYPEDEFINITIONPRT');
030E --      2725                   SYNCHRONIZE; NOTSKIP:=FALSE
0312 -1      2726                END
031E -0 A    2727       END; (* OF TYPEDEFPRT *)
0370 --      2728
0370 -- A    2729       PROCEDURE VARDCLPART;
0040 --      2730       (* <VARIABLE_DECLARATION_PART> *)
0040 --      2731       VAR FIRSTID: HASHPTR; VARIDLST: @IDLST; VARTYP: @TYPES; XRSCNR: INTEGER;
0050 --      2732
0050 -- B    2733       PROCEDURE PUSHVARS;
0040 --      2734       VAR PTRACT: @ACTIVS;
0000 0- B    2735       BEGIN
000A --      2736          WHILE VARIDLST<>NIL DO
0016 1-      2737             BEGIN IF SEARCHONLY(VARIDLST@.ID) THEN
0052 2-      2738                      BEGIN NEW(PTRACT, VARID);
006A --      2739                         WITH PTRACT@ DO
0080 3-      2740                            BEGIN IDNR:=VARIDLST@.ID; LEVNR:=LEVEL; PARM:=VARB;
00C2 --      2741                               NEW(XREF); XREF@.OCC:=XRSCNR; XREF@.NXTREF:=NIL;
0108 --      2742                               TYP:=VARTYP; PUSH(PTRACT)
0144 -3      2743                            END
0148 -2      2744                      END;
0148 --      2745                VARIDLST:=VARIDLST@.NXTID
015C -1      2746             END
017A -0 B    2747       END; (* PUSHVARS *)
0198 --      2748
0198 -- B    2749       PROCEDURE VARDCLPTRM;
0040 --      2750       (* <VARIABLE_DECLARATION_PART_REMAINDER> *)
0000 0- B    2751       BEGIN
000A --      2758          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2759          IF NOTSKIP THEN
001A --      2760          IF TOKENNR IN (.IDENTIFIER,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-      2761             THEN BEGIN
0054 2-      2762                      CASE TOKENNR OF
0062 --      2763                         IDENTIFIER:
0062 3-      2764                            BEGIN FIRSTID:=TOKENFD;
0082 --      2765                               ADVANCE;
0086 --      2766                               IDENTLIST(FIRSTID, VARIDLST);
00AC --      2767                               TERMINAL(COLONSYM,DUMMYFD);
00C2 --      2768                               IF NOT NOTSKIP THEN VARTYP:=UNDFZERO;
00FA --      2769                               XRSCNR:=SCNR;
0102 --      2770                               TYPEPROC(VARTYP, 0); PUSHVARS;
0118 --      2771                               TERMINAL(SEMICSYM,DUMMYFD);
012E --      2772                               VARDCLPTRM
012E -3      2773                            END;
0136 --      2774                         PROCSYM,FUNCSYM,BEGINSYM:
0136 --      2775                            (* EPSILON PRODUCTION *)
0136 -2      2776                      END (* OF CASE *)
```

```
01EC -1      2777                END
01EC 1-      2778             ELSE BEGIN
01F0 --      2779                    ERRMSG26(TOKENNR,'VARIABLEDCLPRTREM');
021A --      2780                    SYNCHRONIZE; NOTSKIP:=FALSE
021E -1      2781                END
022A -0 B    2782       END; (* OF VARDCLPTRM *)
0274 --      2783
0000 0- A    2784       BEGIN
000A --      2785       (* <VARIABLE_DECLARATION_PART> *)
000A --      2792          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2793          IF NOTSKIP THEN
001A --      2794          IF TOKENNR IN (.VARSYM,PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-      2795             THEN BEGIN
0054 2-      2796                     CASE TOKENNR OF
0062 --      2797                        VARSYM:
0062 3-      2798                           BEGIN
0062 --      2799                              ADVANCE;
0066 --      2800                              TERMINAL(IDENTIFIER,FIRSTID);
007A --      2801                              IDENTLIST(FIRSTID, VARIDLST);
00A0 --      2802                              TERMINAL(COLONSYM,DUMMYFD);
00B6 --      2803                              IF NOT NOTSKIP THEN VARTYP:=UNDFZERO;
00EE --      2804                              XRSCNR:=SCNR;
00F6 --      2805                              TYPEPROC(VARTYP, 0); PUSHVARS;
010C --      2806                              TERMINAL(SEMICSYM,DUMMYFD);
0122 --      2807                              VARDCLPTRM
0122 -3      2808                           END;
012A --      2809                        PROCSYM,FUNCSYM,BEGINSYM:
012A --      2810                           (* EPSILON PRODUCTION *)
012A -2      2811                     END (* OF CASE *)
01C4 -1      2812                END
01C4 1-      2813             ELSE BEGIN
01C8 --      2814                     ERRMSG26(TOKENNR,'VARIABLEDECLARPRT');
01F2 --      2815                     SYNCHRONIZE; NOTSKIP:=FALSE
01F6 -1      2816                END
0202 -0 A    2817       END; (* OF VARDCLPART *)
024C --      2818
024C -- A    2819       PROCEDURE FORMALPARM(VAR RETARG: @ARGS; VAR RETACT: @ACTIVS);
0048 --      2820       (* <FORMAL_PARAMETER> *)
0048 --      2821       (* RETARG IS AN ARGUMENT-TYPE-LIST AS REQUIRED FOR ARG *)
0048 --      2822       (* RETACT IS THE LIST OF FORMAL PARAMETER-ID DESCRIPTIONS *)
0048 --      2823       VAR FIRSTID: HASHPTR; PARAMIDLST: @IDLST; PARAMTYP: @TYPES;
0054 --      2824           COMTYPID:HASHPTR; PTRACT: @ACTIVS; VARSW: BOOLEAN;
0060 --      2825           NEWARG, LASTARG: @ARGS;
0000 0- A    2826       BEGIN
000A --      2833          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2834          IF NOTSKIP THEN
001A --      2835          IF TOKENNR IN (.IDENTIFIER,VARSYM,FUNCSYM,PROCSYM.)
001A 1-      2836             THEN BEGIN
0054 2-      2837                     CASE TOKENNR OF
0062 --      2838                        IDENTIFIER:
0062 3-      2839                           BEGIN FIRSTID:=TOKENFD; VARSW:=FALSE;
0088 --      2840                              ADVANCE;
008C --      2841                              IDENTLIST(FIRSTID,PARAMIDLST);
00B2 --      2842                              TERMINAL(COLONSYM,DUMMYFD);
00C8 --      2843                              IF NOTSTAND
00C8 4-      2844                                 THEN BEGIN COMTYPID:=1; (* FOR LATER LOGIC *)
00E0 -4      2845                                      TYPEPROC(PARAMTYP,0) END
00F2 --      2846                                 ELSE TERMINAL(IDENTIFIER,COMTYPID)
```

```
0106 -3      2847                        END;
010E --      2848                     VARSYM:
010E 3-      2849                        BEGIN VARSW:=TRUE;
0116 --      2850                           ADVANCE;
011A --      2851                           TERMINAL(IDENTIFIER,FIRSTID);
012E --      2852                           IDENTLIST(FIRSTID,PARAMIDLST);
0154 --      2853                           TERMINAL(COLONSYM,DUMMYFD);
016A --      2854                           IF NOTSTAND
016A 4-      2855                              THEN BEGIN COMTYPID:=1; (* FOR LATER LOGIC *)
0182 -4      2856                                   TYPEPROC(PARAMTYP,0) END
0194 --      2857                              ELSE TERMINAL(IDENTIFIER,COMTYPID)
01A8 -3      2858                        END;
01B0 --      2859                     FUNCSYM:
01B0 3-      2860                        BEGIN VARSW:=FALSE;
01B6 --      2861                           ERRMSG(398,
01BE --      2862                              'NO FUNCTIONS AS PARAMETERS ALLOWED      ');
01C8 --      2863                           ADVANCE;
01CC --      2864                           TERMINAL(IDENTIFIER,FIRSTID);
01E0 --      2865                           IDENTLIST(FIRSTID,PARAMIDLST);
0206 --      2866                           TERMINAL(COLONSYM,DUMMYFD);
021C --      2867                           TERMINAL(IDENTIFIER,COMTYPID); COMTYPID:=0
0230 -3      2868                        END;
023A --      2869                     PROCSYM:
023A 3-      2870                        BEGIN VARSW:=FALSE;
0240 --      2871                           ERRMSG(398,
0248 --      2872                              'NO PROCEDURES AS PARAMETERS ALLOWED     ');
0252 --      2873                           ADVANCE;
0256 --      2874                           TERMINAL(IDENTIFIER,FIRSTID);
026A --      2875                           IDENTLIST(FIRSTID,PARAMIDLST);
0290 --      2876                           COMTYPID:=0;
0296 -3      2877                        END
0296 -2      2878                  END;(* OF CASE *)
0324 --      2879                  (* COMMON ACTION *)
0324 --      2880                  (* DETERMINE PARAMETERTYPE *)
0324 --      2881                  IF NOTSKIP AND (COMTYPID<>0) AND NOT NOTSTAND
0340 2-      2882                     THEN BEGIN
0356 --      2883                             PTRACT:=SEARCHALL(COMTYPID, TYPID);
039A --      2884                             IF PTRACT@.IDKLASS<>TYPID
03AE 3-      2885                                THEN BEGIN ERRMSG103(COMTYPID);
03D8 -3      2886                                           COMTYPID:=0 END
03DE --      2887                                ELSE PARAMTYP:=PTRACT@.TYP
03F6 -2      2888                          END;
0414 --      2889                  IF NOT NOTSKIP OR (COMTYPID=0)
0434 --      2890                     THEN PARAMTYP:=UNDFZERO;
0460 --      2891                  (* BUILD LISTS TO BE RETURNED *)
0460 --      2892                  RETARG:=NIL; LASTARG:=NIL; RETACT:=NIL;
047A --      2893                  WHILE PARAMIDLST<>NIL DO
0486 2-      2894                     BEGIN IF SEARCHONLY(PARAMIDLST@.ID) THEN
04C2 3-      2895                              BEGIN NEW(PTRACT, VARID);
04DA --      2896                                 WITH PTRACT@ DO
04F0 4-      2897                                    BEGIN IDNR:=PARAMIDLST@.ID;
051E --      2898                                       IF VARSW
051E --      2899                                          THEN PARM:=REFP
0528 --      2900                                          ELSE PARM:=VALP;
053C --      2901                                       LEVNR:=LEVEL; NEW(XREF);
055A --      2902                                       XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL;
0590 --      2903                                       TYP:=PARAMTYP;
05AE --      2904                                       PTRACT@.NXTACT:=RETACT;RETACT:=PTRACT
```

```
05E4 -4      2905                                    END
0606 -3      2906                                 END;
0606 --      2907                              NEW(NEWARG); NEWARG@.ARGTYP:=PARAMTYP;
0648 --      2908                              NEWARG@.PASSKND:=VARSW; NEWARG@.NXTARG:=NIL;
0690 --      2909                              IF LASTARG=NIL
0690 --      2910                                 THEN RETARG:=NEWARG
069C --      2911                                 ELSE LASTARG@.NXTARG:=NEWARG;
06F4 --      2912                              LASTARG:=NEWARG;
0712 --      2913                              PARAMIDLST:=PARAMIDLST@.NXTID
0726 -2      2914                           END
0744 -1      2915                     END
0748 1-      2916                ELSE BEGIN RETARG:=NIL; RETACT:=NIL;
0760 --      2917                        ERRMSG26(TOKENNR,'FORMALPARAMETER  ');
078A --      2918                        SYNCHRONIZE; NOTSKIP:=FALSE
078E -1      2919                     END
079A -0 A    2920           END; (* OF FORMALPARM *)
0840 --      2921
0840 -- A    2922           PROCEDURE FORMPARLST(FIRSTARG: @ARGS; FIRSTACT: @ACTIVS;
0040 --      2923                                VAR CARARGLST: @ARGS; VAR CARACTLST: @ACTIVS);
0050 --      2924           (* <FORMAL_PARAMETER_LIST> *)
0050 --      2925           VAR CURARG, OLDARG, NEWARG, CDRARGLST: @ARGS;
0060 --      2926               CURACT, OLDACT, NEWACT, CDRACTLST: @ACTIVS;
0000 0- A    2927           BEGIN
000A --      2934              (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2935              IF TOKENNR IN (.SEMICSYM,RPARASYM.)
000A 1-      2936                 THEN BEGIN
0048 2-      2937                         CASE TOKENNR OF
0056 --      2938                            SEMICSYM:
0056 3-      2939                               BEGIN
0056 --      2940                                  NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0064 --      2941                                  ADVANCE;
0068 --      2942                                  FORMALPARM(CURARG, CURACT);
007C --      2943                                  FORMPARLST(CURARG, CURACT, CDRARGLST, CDRACTLST)
00C8 -3      2944                               END;
00D0 --      2945                            RPARASYM:
00D0 --      2946                               (* EPSILON PRODUCTION *)
00D0 33      2947                               BEGIN CDRARGLST:=NIL; CDRACTLST:=NIL END
00DC -2      2948                         END;(* OF CASE *)
010A --      2949                         OLDARG:=NIL; OLDACT:=NIL; NEWARG:=FIRSTARG; NEWACT:=FIRSTACT;
0152 --      2950                         WHILE (NEWARG<>NIL) AND (NEWACT<>NIL) DO
017E --      2951                               (* SHOULD ALWAYS TERMINATE SIMULTANOUSLY *)
017E 2-      2952                            BEGIN OLDARG:=NEWARG; NEWARG:=NEWARG@.NXTARG;
01CE --      2953                                  OLDACT:=NEWACT; NEWACT:=NEWACT@.NXTACT;
021E -2      2954                            END;
0222 --      2955                         IF OLDARG<>NIL
0222 22      2956                            THEN BEGIN OLDARG@.NXTARG:=CDRARGLST; CARARGLST:=FIRSTARG END
0282 --      2957                            ELSE CARARGLST:=CDRARGLST;
02A8 --      2958                         IF OLDACT<>NIL
02A8 22      2959                            THEN BEGIN OLDACT@.NXTACT:=CDRACTLST; CARACTLST:=FIRSTACT END
0308 --      2960                            ELSE CARACTLST:=CDRACTLST
030C -1      2961                      END
032E 1-      2962                 ELSE BEGIN
0332 --      2963                         ERRMSG26(TOKENNR,'FORMPARAMETERLIST');
035C --      2964                         GOTO 9999
036A -1      2965                      END
036A -0 A    2966           END; (* OF FORMPARLST *)
03A8 --      2967
03A8 -- A    2968           PROCEDURE FORMPARPRT(VAR ARGLST: @ARGS; VAR ACTLST: @ACTIVS);
```

```
0048 --      2969      (* <FORMAL_PARAMETER_PART> *)
0048 --      2970      VAR CURARG: @ARGS; CURACT: @ACTIVS;
0000 0- A    2971      BEGIN
000A --      2978         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      2979         IF TOKENNR IN (.LPARASYM,SEMICSYM,COLONSYM.)
000A 1-      2980            THEN BEGIN
0048 2-      2981                    CASE TOKENNR OF
0056 --      2982                       LPARASYM:
0056 3-      2983                          BEGIN
0056 --      2984                             ADVANCE;
005A --      2985                             FORMALPARM(CURARG, CURACT);
006E --      2986                             (* NEXT CALL SYNCHRONIZES *)
006E --      2987                             FORMPARLST(CURARG, CURACT, ARGLST, ACTLST);
00BE --      2988                             TERMINAL(RPARASYM,DUMMYFD)
00D0 -3      2989                          END;
00D8 --      2990                       SEMICSYM,COLONSYM:
00D8 3-      2991                          BEGIN
00D8 --      2992                             NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
00E6 --      2993                             (* EPSILON PRODUCTION *)
00E6 --      2994                             ARGLST:=NIL; ACTLST:=NIL
00F0 -3      2995                          END
00FA -2      2996                    END (* OF CASE *)
0144 -1      2997                 END
0144 1-      2998              ELSE BEGIN
0148 --      2999                    ERRMSG26(TOKENNR,'FORMPARAMETERPART');
0172 --      3000                    GOTO 9999
0180 -1      3001                 END
0180 -0 A    3002      END; (* OF FORMPARPRT *)
01C0 --      3003
01C0 -- A    3004      PROCEDURE PROCFUNDCL;
0040 --      3005      (* <PROCEDURE_AND_FUNCTION_DECLARATION> *)
0040 --      3006      VAR PTRACT, ACTLST, RETTYPACT: @ACTIVS; ARGLST: @ARGS; FUNCTYP: @TYPES;
0054 --      3007          ENTRNAME, RETTYPID: HASHPTR; FORWSW, FUNCSW: BOOLEAN; XRSCNR: INTEGER;
0068 --      3008          STMCOD, STMREST, HELPER: CHARVAR;
0000 0- A    3009      BEGIN
000A --      3016         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3017         IF NOTSKIP THEN
001A --      3018         IF TOKENNR IN (.PROCSYM,FUNCSYM.)
001A 1-      3019            THEN BEGIN
0054 2-      3020                    CASE TOKENNR OF
0062 --      3021                       PROCSYM:
0062 3-      3022                          BEGIN ADVANCE; FUNCSW:=FALSE; XRSCNR:=SCNR;
0074 --      3023                             TERMINAL(IDENTIFIER,ENTRNAME); LEVEL:=LEVEL+1;
00A0 --      3024                             IF NOT NOTSKIP THEN GOTO 9999;
00C2 --      3025                             FORMPARPRT(ARGLST, ACTLST);
00D6 --      3026                             TERMINAL(SEMICSYM,DUMMYFD);
00EC --      3027                             FORWSW:=(TOKENNR=IDENTIFIER)AND(TOKENFD=SPFORWARD)
0122 -3      3028                          END;
013E --      3029                       FUNCSYM:
013E 3-      3030                          BEGIN ADVANCE; FUNCSW:=TRUE; XRSCNR:=SCNR;
0152 --      3031                             TERMINAL(IDENTIFIER,ENTRNAME); LEVEL:=LEVEL+1;
017E --      3032                             IF NOT NOTSKIP THEN GOTO 9999;
01A0 --      3033                             FORMPARPRT(ARGLST, ACTLST);
01B4 --      3034                             TERMINAL(COLONSYM,DUMMYFD);
01CA --      3035                             IF NOT NOTSKIP THEN FUNCTYP:=UNDFZERO;
0202 4-      3036                             IF NOTSTAND THEN TYPEPROC(FUNCTYP,0) ELSE BEGIN
0228 --      3037                             RETTYPID:=0;
022E --      3038                             TERMINAL(IDENTIFIER,RETTYPID);
```

```
0242 --  3039                        IF RETTYPID=0
0242 --  3040                           THEN FUNCTYP:=UNDFZERO
024E 5-  3041                           ELSE BEGIN RETTYPACT:=SEARCHALL(RETTYPID,TYPID);
02BA --  3042                                IF RETTYPACT@.IDKLASS=TYPID
02CE --  3043                                   THEN FUNCTYP:=RETTYPACT@.TYP
02EE 6-  3044                                   ELSE BEGIN FUNCTYP:=UNDFZERO;
0334 -6  3045                                              ERRMSG103(RETTYPID) END
0352 -4  3046                           END END;
0352 --  3047                        IF FUNCTYP@.TYPKLASS IN (.ARRTYP, RECTYP.)
0366 4-  3048                           THEN BEGIN
039A --  3049                                ERRDESC:=
039A --  3050                                'FUNC. ''1234567890'': ILLEGAL RESULT TYPE ';
03A6 --  3051                                INSERTWORD:=HASHNAME(.ENTRNAME.);
03CE --  3052                                FOR I:=1 TO 10 DO
03D4 --  3053                                    ERRDESC(.7+I.):=INSERTWORD(.I.);
044E --  3054                                ERRMSG(120,ERRDESC);
0466 --  3055                                FUNCTYP:=UNDFZERO
0466 -4  3056                           END;
048A --  3057                     TERMINAL(SEMICSYM,DUMMYFD);
04A0 --  3058                     FORWSW:=(TOKENNR=IDENTIFIER)AND(TOKENFD=SPFORWARD)
04D6 -3  3059                  END
04EE -2  3060               END;(* OF CASE *)
053C --  3061               IF FORWSW
053C 2-  3062                  THEN BEGIN LEVEL:=LEVEL-1; ADVANCE;
0562 3-  3063                        IF SEARCHONLY(ENTRNAME) THEN BEGIN
058A --  3064                           IF FUNCSW THEN NEW(PTRACT, FUNCID)
05AC --  3065                                     ELSE NEW(PTRACT, PROCID);
05C8 4-  3066                           WITH PTRACT@ DO BEGIN
05DE --  3067                              IDNR:=ENTRNAME; LEVNR:=LEVEL;
0606 --  3068                              NEW(XREF); XREF@.OCC:=XRSCNR;
0632 --  3069                              XREF@.NXTREF:=NIL; SWFORW:=TRUE;
0654 --  3070                              ARG:=ARGLST; FORWARGS:=ACTLST;
0690 --  3071                              IF FUNCSW THEN RETTYP:=FUNCTYP
069A -4  3072                                        ELSE RETTYP:=UNDFZERO END;
06E0 -3  3073                           PUSH(PTRACT) END
0702 -2  3074                     END
0702 2-  3075                  ELSE BEGIN LEVEL:=LEVEL-1;
071E 3-  3076                        IF SEARCHONLY(ENTRNAME) OR NOT NOTSKIP THEN BEGIN
0756 --  3077                           (* NO ENTRY FOR THIS IDENTIFIER OR ERRORMODE *)
0756 --  3078                           IF FUNCSW THEN NEW(PTRACT, FUNCID)
0778 --  3079                                     ELSE NEW(PTRACT, PROCID);
0794 4-  3080                           WITH PTRACT@ DO BEGIN
07AA --  3081                              IDNR:=ENTRNAME; LEVNR:=LEVEL;
07D2 --  3082                              NEW(XREF); XREF@.OCC:=XRSCNR;
07FE --  3083                              XREF@.NXTREF:=NIL; SWFORW:=FALSE;
081E --  3084                              ARG:=ARGLST;
083C --  3085                              IF FUNCSW THEN RETTYP:=FUNCTYP
0846 -4  3086                                        ELSE RETTYP:=UNDFZERO END;
088C --  3087                           PUSH(PTRACT)
08AA 33  3088                           END ELSE BEGIN (*LOOK FOR FORWARD REFERENCE*)
08B2 --  3089                           (* IF NOT FORWARD, SEARCHONLY PRINTED MESSAGE *)
08B2 --  3090                           PTRACT:=SRCHTAB(.ENTRNAME.)@.ACTLSTPTR;
0904 --  3091                           IF PTRACT@.IDKLASS IN (.PROCID,FUNCID.) THEN
094C 4-  3092                              IF PTRACT@.SWFORW THEN BEGIN
096A --  3093                                 (* IT IS A FORWARD REFERENCE *)
096A --  3094                                 IF ACTLST<>NIL THEN
0976 --  3095                                    ERRMSG(119,
097E --  3096                                     'FORW. DCL.:MUST NOT REPEAT ARGUMENT LIST');
```

```
0988 --      3097                                        PTRACT@.SWFORW:=FALSE; (* NOT AGAIN *)
09A2 -4      3098                                        ACTLST:=PTRACT@.FORWARGS END;
09D4 -3      3099                                   END; (* ACTLST CONTAINS FORMAL PARAMETERS *)
09D4 --      3100                                   (* PTRACT CONTAINS ID-RECORD (ALSO IF ANY ERROR) *)
09D4 --      3101                                   LEVEL:=LEVEL+1;
09EC --      3102                                   (* LOCATE CODE GENERATION VARIABLES *)
09EC --      3103                                   LOC(HELPER, 15); LOC(STMREST, 50);
0A14 --      3104                                   LOC(STMCOD, 200);
0A28 --      3105                                   STMREST@.CURLG:=0;
0A42 --      3106                                   CODIDENT(PTRACT, STMCOD); (*NOW, IT WILL BE REUSED!*)
0A6C --      3107                                   WHILE ACTLST<>NIL DO
0A78 3-      3108                                      BEGIN PTRACT:=ACTLST;
0A96 --      3109                                         ACTLST:=ACTLST@.NXTACT; PUSH(PTRACT);
0AEA --      3110                                         CODIDENT(PTRACT, HELPER);
0B14 --      3111                                         CAT(HELPER, VALPREF, HELPER);
0B62 --      3112                                         IF STMREST@.CURLG<>0
0B76 --      3113                                            THEN APP(STMREST, LITCOMMA)
0BAE --      3114                                            ELSE LIT(STMREST, 2, '@ 3456789012345');
0BD0 --      3115                                         APP(STMREST, HELPER)
0BF6 -3      3116                                      END;
0BFE --      3117                                   IF STMREST@.CURLG<>0
0C12 --      3118                                      THEN APP(STMREST, LITCOLON);
0C4E --      3119                                   CODINTEG(STMNR+1, HELPER);
0C6C --      3120                                   CAT(HELPER, STM, HELPER);
0CBA --      3121                                   APP(STMREST, HELPER);
0CE4 --      3122                                   IF STMREST@.CURLG<>HELPER@.CURLG
0D0C --      3123                                      THEN WRAP('(',STMREST,')');
0D34 --      3124                                   FREE(HELPER);
0D56 --      3125                                   APP(STMCOD, EQUIV);
0D86 --      3126                                   APP(STMCOD, STMREST); FREE(STMREST);
0DD2 --      3127                                   APP(STMCOD, TERMIN);
0E02 --      3128                                   OUT(STMCOD); FREE(STMCOD);
0E46 --      3129                                   BLOCK;
0E4A --      3130                                   POP; LEVEL:=LEVEL-1
0E4E -2      3131                                END
0E66 -1      3132                          END
0E66 1-      3133                       ELSE BEGIN
0E6A --      3134                             ERRMSG26(TOKENNR,'PROCFUNCDECLARAT ');
0E94 --      3135                             SYNCHRONIZE; NOTSKIP:=FALSE
0E98 -1      3136                          END
0EA4 -0 A    3137                END; (* OF PROCFUNDCL *)
0F6C --      3138
0F6C -- A    3139                PROCEDURE PROCFUNPRT;
0040 --      3140                (* <PROCEDURE_AND_FUNCTION_DECLARATION_PART> *)
0000 0- A    3141                BEGIN
000A --      3148                   (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3149                   IF NOTSKIP THEN
001A --      3150                   IF TOKENNR IN (.PROCSYM,FUNCSYM,BEGINSYM.)
001A 1-      3151                      THEN BEGIN
0054 2-      3152                             CASE TOKENNR OF
0062 --      3153                                PROCSYM,FUNCSYM:
0062 3-      3154                                   BEGIN
0062 --      3155                                      PROCFUNDCL;
0066 --      3156                                      TERMINAL(SEMICSYM,DUMMYFD);
007C --      3157                                      PROCFUNPRT
007C -3      3158                                   END;
0084 --      3159                                BEGINSYM:
0084 --      3160                                   (* EPSILON PRODUCTION *)
```

```
0084 -2      3161                    END (* OF CASE *)
00FA -1      3162                 END
00FA 1-      3163              ELSE BEGIN
00FE --      3164                    ERRMSG26(TOKENNR,'PROCFUNCDCLPART  ');
0128 --      3165                    SYNCHRONIZE; NOTSKIP:=FALSE
012C -1      3166                 END
0138 -0 A    3167        END; (* OF PROCFUNPRT *)
0174 --      3168
0174 -- A    3169        PROCEDURE EXPRESLIST(FIRSTCOD: CHARVAR; FIRSTTYP: @TYPES; FIRSTSWVAR: BOOLEAN;
0040 --      3170                             VAR CAREXPLST: @EXPLST);
0050 --      3171        (* <EXPRESSION_LIST> *)
0050 --      3172        VAR CUREXPCOD: CHARVAR; CUREXPTYP: @TYPES; CURSWVAR: BOOLEAN;
005C --      3173            CDREXPLST: @EXPLST;
0000 0- A    3174        BEGIN
000A --      3181           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3182           IF NOTSKIP THEN
001A --      3183           IF TOKENNR IN (.COMMASYM,RPARASYM,RBRACKSYM.)
001A 1-      3184              THEN BEGIN
0058 2-      3185                      CASE TOKENNR OF
0066 --      3186                         COMMASYM:
0066 3-      3187                            BEGIN
0066 --      3188                               ADVANCE;
006A --      3189                               LOC(CUREXPCOD, 50); (* DON'T FREE IT ON RETURN *)
007E --      3190                               EXPRESSION(CUREXPCOD, CUREXPTYP);
0092 --      3191                               IF CUREXPTYP@.TYPKLASS<>UNDEF
00A6 --      3192                                  THEN CURSWVAR:=TOPBOOL@.UP@.SWVAR
00E0 --      3193                                  ELSE CURSWVAR:=TRUE;
0106 --      3194                               EXPRESLIST(CUREXPCOD, CUREXPTYP, CURSWVAR, CDREXPLST);
0168 -3      3195                            END;
016C --      3196                         RPARASYM,RBRACKSYM:
016C --      3197                            (* EPSILON PRODUCTION *)
016C --      3198                            CDREXPLST:=NIL
016C -2      3199                      END; (* OF CASE *)
01A4 --      3200                      NEW(CAREXPLST);
01B8 --      3201                      CAREXPLST@.EXPLSTCOD:=FIRSTCOD;
01EE --      3202                      CAREXPLST@.EXPLSTTYP:=FIRSTTYP;
0224 --      3203                      CAREXPLST@.SWVAR:=FIRSTSWVAR;
0256 --      3204                      CAREXPLST@.NXTEXP:=CDREXPLST
026E -1      3205                   END
028C 1-      3206              ELSE BEGIN
0290 --      3207                      ERRMSG26(TOKENNR,'EXPRESSIONLIST   ');
02BA --      3208                      SYNCHRONIZE; NOTSKIP:=FALSE; CAREXPLST:=NIL
02CA -1      3209                   END
02D4 --      3210           ELSE CAREXPLST:=NIL (* EXCEPTION FOR ERRORMODE: RETURN NIL, TOO *)
02D8 -0 A    3211        END; (* OF EXPRESLIST *)
0330 --      3212
0330 -- A    3213        PROCEDURE VARSELECTS(SWASSIGN: BOOLEAN; (* TRUE IF LEFT SIDE OF ASSIGNMENT *)
0040 --      3214                             VAR REPLLST: @SUBSBNDS; (* LIST OF SUBSCRIPTS AND ARRAY  *)
0040 --      3215                                                     (* BOUNDS IN CASE OF ANASSIGNMENT*)
0040 --      3216                             CODIN: CHARVAR; TYPIN: @TYPES;
0040 --      3217                             VAR CODOUT: CHARVAR; VAR TYPOUT: @TYPES);
0058 --      3218        (* <VARIABLE_SELECTORS> *)
0058 --      3219        VAR FIRSTCOD: CHARVAR; FIRSTTYP: @TYPES; INDXEXPLST: @EXPLST;
0064 --      3220            SHIFTNR, EXTENSION, LITMINUS, SH (* USED AS PTR SHORTHAND *): CHARVAR;
0074 --      3221            LASTRL: @SUBSBNDS; (* LAST ELEMENT IN REPLLST *)
0078 --      3222            ID: HASHPTR; SECLST, OLDSEL: @ACTIVS; NEWX: @XREFS; SIGNAL: BOOLEAN;
0000 0- A    3223        BEGIN
000A --      3224           LOC(SHIFTNR,3); LOC(EXTENSION,3);
```

```
0032 --      3225          LOC(LITMINUS,2); LIT(LITMINUS,7,'$MINUS 89012345');
0060 --      3232          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0060 --      3233          IF NOTSKIP THEN
0070 --      3234          IF TOKENNR IN (.LBRACKSYM,PERIODSYM,POINTER,RPARASYM,SEMICSYM,
0070 --      3235                         COMMASYM,EQUALSYM,RBRACKSYM,DOUBLEDOT,BECOMES,DOSYM,
0070 --      3236                         OFSYM,ORSYM,TOSYM,ENDSYM,ELSESYM,THENSYM,UNTILSYM,
0070 --      3237                         DOWNTOSYM,PLUSMINUS,RELOPER,MULTOPER.)
0070 1-      3238             THEN BEGIN
00AA 2-      3239                     CASE TOKENNR OF
00B8 --      3240                        LBRACKSYM:
00B8 3-      3241                           BEGIN
00B8 --      3242                              ADVANCE;
00BC --      3243                              LOC(FIRSTCOD, 50); (* DON'T FREE ON RETURN *)
00D0 --      3244                              EXPRESSION(FIRSTCOD, FIRSTTYP);
00E4 --      3245                              EXPRESLIST(FIRSTCOD, FIRSTTYP, TRUE, INDXEXPLST);
0134 --      3246                              IF SWASSIGN THEN
013E 4-      3247                                 BEGIN (* OBTAIN LASTRL *)
013E --      3248                                    LASTRL:=REPLLST;
0160 --      3249                                    IF LASTRL<>NIL THEN
016C --      3250                                       WHILE LASTRL@.NXTSBND<>NIL DO
018C --      3251                                          LASTRL:=LASTRL@.NXTSBND
01A0 -4      3252                                 END;
01C2 4-      3253                              WHILE INDXEXPLST<>NIL DO BEGIN
01CE --      3254                                 IF TYPIN@.TYPKLASS<>UNDEF THEN
01EE --      3255                                    IF TYPIN@.TYPKLASS<>ARRTYP
0202 5-      3256                                       THEN BEGIN ERRMSG(138,
0216 --      3257                                            'TYPE OF VARIABLE IS NOT AN ARRAY        ');
0220 -5      3258                                            TYPIN:=UNDFZERO END
0244 5-      3259                                       ELSE BEGIN
0248 --      3260                                            COMPARE(TYPIN@.INDXTYP,
027A --      3261                                                    INDXEXPLST@.EXPLSTTYP);
02B0 --      3262                                            SH:=INDXEXPLST@.EXPLSTCOD;
02E2 --      3263                                            IF TYPIN@.INDXTYP@.TYPKLASS=SCALTYP THEN
0316 6-      3264                                               BEGIN
0316 --      3265                                                  IF TYPIN@.INDXTYP@.LOWBND<>1
033E --      3266                                                     THEN IF TYPIN@.INDXTYP=TYPBOOLEAN
035E 7-      3267                                                          THEN BEGIN
0370 --      3268                                                             LIT(SHIFTNR, 2,
0380 --      3269                                                                 ' 134567890123456789');
038A --      3270                                                             APP(SH, SHIFTNR);
03B4 --      3271                                                             WRAP('(', SH, ')');
03D0 --      3272                                                             LIT(SHIFTNR,1,
03E0 --      3273                                                             '2234567890123456789');
03EA --      3274                                                             APP(SH, SHIFTNR);
0414 --      3275                                                             WRAP('(', SH, ')')
042C -7      3276                                                          END
0430 7-      3277                                                          ELSE BEGIN
0434 --      3278                                                            CAT(SH, LITMINUS, SH);
047C --      3279                                                            WRAP('(',SH,')');
0498 --      3280                                                            CODINTEG(
0498 --      3281                                                              TYPIN@.INDXTYP@.LOWBND-1,
04CC --      3282                                                              SHIFTNR);
04D8 --      3283                                                            APP(SH, SHIFTNR);
0502 --      3284                                                            WRAP('(',SH,')')
051A -7      3285                                                          END;
051E --      3286                                                  CODINTEG(TYPIN@.INDXTYP@.HIGBND-
0546 --      3287                                                           TYPIN@.INDXTYP@.LOWBND+1,
057E --      3288                                                           EXTENSION);
```

```
058A -6      3289                                        END;
058A --      3290                                     IF SWASSIGN
058A 6-      3291                                        THEN BEGIN
0594 --      3292                                             IF LASTRL<>NIL
0594 7-      3293                                                THEN BEGIN
05A0 --      3294                                                     NEW(LASTRL@.NXTSBND);
05C4 --      3295                                                     LASTRL:=LASTRL@.NXTSBND
05D8 -7      3296                                                     END
05F6 7-      3297                                                ELSE BEGIN
05FA --      3298                                                     NEW(REPLLST);
060E --      3299                                                     LASTRL:=REPLLST
060E -7      3300                                                     END;
0630 --      3301                                             LASTRL@.SUB:=SH;(*OVERLAY STRINGS*)
0662 --      3302                                             LOC(LASTRL@.BND, 3);
068A --      3303                                             SSS(LASTRL@.BND, EXTENSION);
06C8 --      3304                                             LASTRL@.NXTSBND:=NIL
06DC -6      3305                                             END
06E2 6-      3306                                        ELSE BEGIN
06E6 --      3307                                             CAT(SH, RETRIEVE, SH);
0734 --      3308                                             WRAP('(',SH,')');
0750 --      3309                                             APP(SH, EXTENSION);
077A --      3310                                             WRAP('(',SH,')');
0796 --      3311                                             APP(CODIN, SH); FREE(SH);
07E2 --      3312                                             WRAP('(',CODIN,')');
07FE -6      3313                                             END;
07FE --      3314                                     TYPIN:=TYPIN@.COMPTYP;
0830 -5      3315                                     END;
0830 -4      3316                                INDXEXPLST:=INDXEXPLST@.NXTEXP END;
0866 --      3317                             TERMINAL(RBRACKSYM,DUMMYFD);
087C --      3318                             VARSELECTS(SWASSIGN,REPLLST, CODIN,TYPIN, CODOUT,TYPOUT)
08EA -3      3319                          END;
08F2 --      3320                       PERIODSYM:
08F2 3-      3321                          BEGIN
08F2 --      3322                             ADVANCE;
08F6 --      3323                             TERMINAL(IDENTIFIER, ID);
090A --      3324                             IF TYPIN@.TYPKLASS<>UNDEF THEN
092A --      3325                                IF TYPIN@.TYPKLASS<>RECTYP
093E 4-      3326                                   THEN BEGIN ERRMSG(140,
0952 --      3327                                    'TYPE OF VARIABLE IS NOT A RECORD        ');
095C -4      3328                                         TYPIN:=UNDFZERO END
0980 4-      3329                                   ELSE BEGIN SECLST:=TYPIN@.SECTNS;
09B6 --      3330                                          SIGNAL:=(ID<>0); OLDSEL:=NIL;
09E4 --      3331                                          WHILE (SECLST<>NIL) AND SIGNAL DO
0A00 5-      3332                                             BEGIN SIGNAL:=SECLST@.IDNR<>ID;
0A3C --      3333                                                OLDSEL:=SECLST;
0A5A -5      3334                                                SECLST:=SECLST@.NXTACT END;
0A90 5-      3335                                          IF SIGNAL THEN BEGIN
0A9A --      3336                                             ERRDESC:=
0A9A --      3337                                         'RECORD FIELD ''1234567890'' NOT FOUND     ';
0AA6 --      3338                                             INSERTWORD:=HASHNAME(.ID.);
0ACE --      3339                                             FOR I:=1 TO 10 DO
0AD4 --      3340                                             ERRDESC(.14+I.):=INSERTWORD(.I.);
0B4E --      3341                                             ERRMSG(152, ERRDESC);
0B66 -5      3342                                             TYPIN:=UNDFZERO END
0B8A --      3343                                             ELSE
0B8E --      3344                                             IF OLDSEL<>NIL
0B8E 5-      3345                                                THEN BEGIN TYPIN:=OLDSEL@.TYP;
0BCC --      3346                                                     NEW(NEWX); NEWX@.OCC:=SCNR;
```

```
0BF8 --      3347                                                NEWX@.NXTREF:=OLDSEL@.XREF;
0C3E -5      3348                                                OLDSEL@.XREF:=NEWX END
0C70 --      3349                                            ELSE TYPIN:=UNDFZERO
0C74 -4      3350                                        END;
0C98 --      3351                                VARSELECTS(SWASSIGN,REPLLST, CODIN,TYPIN, CODOUT,TYPOUT)
0D06 -3      3352                             END;
0D0E --      3353                          POINTER:
0D0E 3-      3354                             BEGIN
0D0E --      3355                                ADVANCE;
0D12 --      3356                                IF TYPIN@.TYPKLASS<>UNDEF THEN
0D32 --      3357                                   IF TYPIN@.TYPKLASS<>PTRTYP
0D46 4-      3358                                      THEN BEGIN ERRMSG(141,
0D5A --      3359                                             'VARIABLE IS NOT A POINTER              ');
0D64 -4      3360                                              TYPIN:=UNDFZERO END
0D88 --      3361                                      ELSE TYPIN:=TYPIN@.REFERTYP;
0DBE --      3362                                VARSELECTS(TRUE, DUMMYSBND, CODIN, TYPIN, CODOUT, TYPOUT)
0E1C -3      3363                             END;
0E24 --      3364                          RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,RBRACKSYM,DOUBLEDOT,
0E24 --      3365                          BECOMES,DOSYM,OFSYM,ORSYM,TOSYM,ENDSYM,ELSESYM,THENSYM,
0E24 --      3366                          UNTILSYM,DOWNTOSYM,PLUSMINUS,RELOPER,MULTOPER:
0E24 --      3367                             (* EPSILON PRODUCTION *)
0E24 3-      3368                             BEGIN
0E24 --      3369                                TYPOUT:=TYPIN; SSS(CODOUT, CODIN)
0E6C -3      3370                             END
0E70 -2      3371                       END (* OF CASE *)
0F72 -1      3372                   END
0F72 1-      3373               ELSE BEGIN
0F76 --      3374                       ERRMSG26(TOKENNR,'VARIABLESELECTOR ');
0FA0 --      3375                       SYNCHRONIZE; NOTSKIP:=FALSE; TYPOUT:=UNDFZERO
0FB0 -1      3376                   END
0FD8 --      3377            ELSE TYPOUT:=UNDFZERO; (* ALSO IN ERROR MODE *)
1004 --      3378            FREE(SHIFTNR); FREE(EXTENSION); FREE(LITMINUS);
106A -0 A    3379         END; (* OF VARSELECTS *)
11A8 --      3380
11A8 -- A    3381         PROCEDURE ACTPARAMPT(VAR ACTEXPLST: @EXPLST);
0044 --      3382         (* <ACTUAL_PARAMETER_PART> *)
0044 --      3383         VAR FIRSTCOD: CHARVAR; FIRSTTYP: @TYPES; FIRSTSWVAR: BOOLEAN;
0000 0- A    3384         BEGIN
000A --      3391            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3392            IF NOTSKIP THEN
001A --      3393            IF TOKENNR=LPARASYM
001A 1-      3394               THEN BEGIN
002C --      3395                       ADVANCE;
0030 --      3396                       LOC(FIRSTCOD, 50); (* DON'T FREE ON RETURN *)
0044 --      3397                       EXPRESSION(FIRSTCOD, FIRSTTYP);
0058 --      3398                       IF FIRSTTYP@.TYPKLASS<>UNDEF
006C --      3399                          THEN FIRSTSWVAR:=TOPBOOL@.UP@.SWVAR
00A6 --      3400                          ELSE FIRSTSWVAR:=TRUE;
00CC --      3401                       EXPRESLIST(FIRSTCOD, FIRSTTYP, FIRSTSWVAR, ACTEXPLST);
012E --      3402                       TERMINAL(RPARASYM,DUMMYFD)
0140 -1      3403                   END
0144 1-      3404               ELSE BEGIN
0148 --      3405                       ERRMSG26(TOKENNR,'ACTPARAMETERPART ');
0172 --      3406                       SYNCHRONIZE; NOTSKIP:=FALSE; ACTEXPLST:=NIL
0182 -1      3407                   END
018C --      3408            ELSE ACTEXPLST:=NIL (* ALSO IN ERRORMODE *)
0190 -0 A    3409         END; (* OF ACTPARAMPT *)
01D4 --      3410
```

```
01D4 -- A   3411      PROCEDURE IDENTREM(ID: HASHPTR; VAR CODOUT: CHARVAR; VAR TYPOUT: @TYPES);
004C --     3412      (* <IDENTIFIER_REMAINDER> *)
004C --     3413      VAR CODIN: CHARVAR; TYPIN: @TYPES;
0054 --     3414          WORD: CHARVAR; (* WILL HOLD 'CHR ' *)
0058 --     3415          PTRACT: @ACTIVS; ACTEXPLST: @EXPLST; ARGLST: @ARGS;
0064 --     3416          RUNACT: @EXPLST; RUNARG, OLDARG: @ARGS;
0000 0- A   3417      BEGIN
000A --     3418         LOC(CODIN, 50);
001E --     3425         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --     3426         IF NOTSKIP THEN
002E --     3427         IF TOKENNR IN (.PERIODSYM,RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,
002E --     3428                        LBRACKSYM,RBRACKSYM,DOUBLEDOT,DOSYM,OFSYM,ORSYM,
002E --     3429                        TOSYM,ENDSYM,ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM,
002E --     3430                        POINTER,PLUSMINUS,RELOPER,MULTOPER,LPARASYM.)
002E 1-     3431            THEN BEGIN
0068 2-     3432                    CASE TOKENNR OF
0076 --     3433                       PERIODSYM,RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,LBRACKSYM,
0076 --     3434                       RBRACKSYM,DOUBLEDOT,DOSYM,OFSYM,ORSYM,TOSYM,ENDSYM,
0076 --     3435                       ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM,POINTER,PLUSMINUS,
0076 --     3436                       RELOPER,MULTOPER:
0076 3-     3437                          BEGIN
0076 --     3438                             PTRACT:=SEARCHALL(ID, VARID);
00BA 4-     3439                             CASE PTRACT@.IDKLASS OF
00D6 --     3440                             SCALID:
00D6 5-     3441                                BEGIN TOPBOOL@.SWVAR:=FALSE;
00F6 --     3442                                   TYPIN:=PTRACT@.STYP;
0128 --     3443                                   IF TYPIN@.TYPKLASS=SCALTYP
013C --     3444                                      THEN IF TYPIN=TYPCHAR
0148 6-     3445                                              THEN BEGIN
015A --     3446                                                      CODINTEG(PTRACT@.VALUU, CODIN);
0182 --     3447                                                      LOC(WORD,5);
0196 --     3448                                                      LIT(WORD,4,'CHR 56789012345');
01B0 --     3449                                                      CAT(CODIN, WORD, CODIN);
01F8 --     3450                                                      FREE(WORD); WRAP('(',CODIN,')')
0232 -6     3451                                                   END
0236 --     3452                                             ELSE
023A --     3453                                             IF TYPIN=TYPBOOLEAN
023A --     3454                                             THEN IF PTRACT@.VALUU=0
0260 --     3455                                                  THEN LIT(CODIN,5,'$TRUE6789012345')
0282 --     3456                                                  ELSE LIT(CODIN,6,'$FALSE789012345')
02A0 --     3457                                             ELSE CODINTEG(PTRACT@.VALUU, CODIN)
02CC --     3458                                      ELSE; (* STRING AS CONSTANT, NOT YET *)
02D4 -5     3459                                END;
02D8 --     3460                             TYPID,PROCID:
02D8 55     3461                                BEGIN ERRMSG103(ID); TYPIN:=UNDFZERO END;
031E --     3462                             FLDID,VARID:
031E 5-     3463                                BEGIN TYPIN:=PTRACT@.TYP;
0350 --     3464                                   CODIDENT(PTRACT, CODIN);
037A -5     3465                                END;
037E --     3466                             FUNCID:
037E 5-     3467                                BEGIN TOPBOOL@.SWVAR:=FALSE;
039E --     3468                                   (* MUST BE WITHOUT PARAMETERS *)
039E --     3469                                   IF PTRACT@.ARG<>NIL THEN
03BE --     3470                                      ERRMSG(126,
03C6 --     3471                                          'ACTUAL NUMBER OF ARGUMENTS UNEQUALS DCL.');
03D0 --     3472                                   TYPIN:=PTRACT@.RETTYP
03E4 -5     3473                                END
0402 -4     3474                             END; (*CASE*)
```

```
0438 --    3475                        VARSELECTS(FALSE, DUMMYSBND,
0448 --    3476                                   CODIN, TYPIN, CODOUT, TYPOUT);
0498 -3    3477                     END;
049C --    3478                  LPARASYM:
049C 3-    3479                     BEGIN TOPBOOL@.SWVAR:=FALSE;
04BC --    3480                        PTRACT:=SEARCHALL(ID, FUNCID);
0500 --    3481                        IF PTRACT@.IDKLASS<>FUNCID
0514 4-    3482                           THEN BEGIN ERRMSG103(ID); ARGLST:=NIL;
0544 -4    3483                                      TYPOUT:=UNDFZERO END
056C 4-    3484                           ELSE BEGIN ARGLST:=PTRACT@.ARG;
05A2 -4    3485                                      TYPOUT:=PTRACT@.RETTYP END;
05D8 --    3486                        ACTPARAMPT(ACTEXPLST);
05E4 --    3487                        (* CODE FOR PREDEFINED FUNCTION "ORD" *)
05E4 --    3488                        IF (PTRACT@.IDNR=SPORD) AND (PTRACT@.LEVNR=0) THEN
063E --    3489                           IF ACTEXPLST<>NIL THEN CODOUT:=ACTEXPLST@.EXPLSTCOD;
0680 --    3490                        (* CODE FOR PREDEFINED FUNCTION "CHR", "PRED", "SUCC" *)
0680 --    3491                        IF ((PTRACT@.IDNR=SPCHR) OR (PTRACT@.IDNR=SPPRED) OR
06DA --    3492                            (PTRACT@.IDNR=SPSUCC)) AND (PTRACT@.LEVNR=0) THEN
0736 --    3493                           IF ACTEXPLST<>NIL THEN
0742 4-    3494                              BEGIN CODIDENT(PTRACT, CODOUT);
076C --    3495                                 APP(CODOUT, LITBLANK);
079C --    3496                                 APP(CODOUT, ACTEXPLST@.EXPLSTCOD);
07DA --    3497                                 WRAP('(',CODOUT,')')
07F2 -4    3498                              END;
07F6 --    3499                        IF TYPOUT=NIL THEN
0806 4-    3500                           BEGIN (* TAKE ACTUAL ONE *)
0806 --    3501                              RUNACT:=ACTEXPLST;
0824 --    3502                              WHILE RUNACT<>NIL DO
0830 5-    3503                                 BEGIN NEW(RUNARG);
0840 --    3504                                    IF ARGLST=NIL
0840 --    3505                                       THEN ARGLST:=RUNARG
084C --    3506                                       ELSE OLDARG@.NXTARG:=RUNARG;
08A0 --    3507                                    RUNARG@.ARGTYP:=RUNACT@.EXPLSTTYP;
08E6 --    3508                                    RUNARG@.PASSKND:=FALSE; OLDARG:=RUNARG;
091E --    3509                                    RUNACT:=RUNACT@.NXTEXP
0932 -5    3510                                 END; OLDARG@.NXTARG:=NIL;
096E --    3511                                 PTRACT@.RETTYP:=UNDFZERO;
09A6 --    3512                                 PTRACT@.ARG:=ARGLST; PUSH(PTRACT);
09FA --    3513                                 TYPOUT:=UNDFZERO;
0A22 -4    3514                           END;
0A22 --    3515                        WHILE (ACTEXPLST<>NIL) AND (ARGLST<>NIL) DO
0A4E 4-    3516                           BEGIN COMPARE(ACTEXPLST@.EXPLSTTYP,
0A80 --    3517                                         ARGLST@.ARGTYP);
0AB6 --    3518                              IF NOT ACTEXPLST@.SWVAR
0ACA --    3519                                 AND ARGLST@.PASSKND THEN
0AF0 --    3520                                 ERRMSG(154,
0AF8 --    3521                                    'ACTUAL PARAMETER MUST BE A VARIABLE     ');
0B02 --    3522                              ACTEXPLST:=ACTEXPLST@.NXTEXP;
0B34 -4    3523                              ARGLST:=ARGLST@.NXTARG END;
0B6A --    3524                        IF (ACTEXPLST<>NIL) OR (ARGLST<>NIL)
0B8E --    3525                           THEN ERRMSG(126,
0B9E --    3526                                       'ACTUAL NUMBER OF ARGUMENTS UNEQUALS DCL.')
0BA4 -3    3527                     END
0BA8 -2    3528                  END (* OF CASE *)
0CAA -1    3529               END
0CAA 1-    3530            ELSE BEGIN
0CAE --    3531                  ERRMSG26(TOKENNR,'IDENTIFIERREMAIND');
0CD8 --    3532                  SYNCHRONIZE; NOTSKIP:=FALSE; TYPOUT:=UNDFZERO;
```

```
0D10 -1      3533                END
0D10 --      3534             ELSE TYPOUT:=UNDFZERO;
0D3C --      3535             FREE(CODIN)
0D5A -0 A    3536          END; (* OF IDENTREM *)
0E2C --      3537
0E2C -- A    3538          PROCEDURE SETELEMREM;
0040 --      3539          (* <SET_ELEMENT_REMAINDER> *)
0000 0- A    3540          BEGIN
000A --      3547             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3548             IF NOTSKIP THEN
001A --      3549             IF TOKENNR IN (.DOUBLEDOT,COMMASYM,RBRACKSYM.)
001A 1-      3550                THEN BEGIN
0058 2-      3551                        CASE TOKENNR OF
0066 --      3552                           DOUBLEDOT:
0066 3-      3553                              BEGIN
0066 --      3554                                 ADVANCE;
006A --      3555                                 EXPRESSION(DUMMYCOD,DUMMYTYP)
007E -3      3556                              END;
0086 --      3557                           COMMASYM,RBRACKSYM:
0086 --      3558                              (* EPSILON PRODUCTION *)
0086 -2      3559                        END (* OF CASE *)
00C6 -1      3560                     END
00C6 1-      3561                ELSE BEGIN
00CA --      3562                        ERRMSG26(TOKENNR,'SETELEMENTREMAIND');
00F4 --      3563                        SYNCHRONIZE; NOTSKIP:=FALSE
00F8 -1      3564                     END
0104 -0 A    3565          END; (* OF SETELEMREM *)
014C --      3566
014C -- A    3567          PROCEDURE SETELEMENT;
0040 --      3568          (* <SET_ELEMENT> *)
0000 0- A    3569          BEGIN
000A --      3576             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3577             IF NOTSKIP THEN
001A --      3578             IF TOKENNR IN (.LPARASYM,LBRACKSYM,NILSYM,NOTSYM,STRINGSYM,
001A --      3579                            PLUSMINUS,IDENTIFIER,UNSGREAL,UNSGINTEG.)
001A 1-      3580                THEN BEGIN
0054 --      3581                        EXPRESSION(DUMMYCOD,DUMMYTYP);
006C --      3582                        SETELEMREM
006C -1      3583                     END
0070 1-      3584                ELSE BEGIN
0074 --      3585                        ERRMSG26(TOKENNR,'SETELEMENT       ');
009E --      3586                        SYNCHRONIZE; NOTSKIP:=FALSE
00A2 -1      3587                     END
00AE -0 A    3588          END; (* OF SETELEMENT *)
00E8 --      3589
00E8 -- A    3590          PROCEDURE SETELEMLST;
0040 --      3591          (* <SET_ELEMENT_LIST> *)
0000 0- A    3592          BEGIN
000A --      3599             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3600             IF NOTSKIP THEN
001A --      3601             IF TOKENNR IN (.COMMASYM,RBRACKSYM.)
001A 1-      3602                THEN BEGIN
0058 2-      3603                        CASE TOKENNR OF
0066 --      3604                           COMMASYM:
0066 3-      3605                              BEGIN
0066 --      3606                                 ADVANCE;
006A --      3607                                 SETELEMENT;
006E --      3608                                 SETELEMLST
```

```
006E -3      3609                        END;
0076 --      3610                     RBRACKSYM:
0076 --      3611                        (* EPSILON PRODUCTION *)
0076 -2      3612                  END (* OF CASE *)
00A6 -1      3613               END
00A6 1-      3614            ELSE BEGIN
00AA --      3615                  ERRMSG26(TOKENNR,'SETELEMENTLIST   ');
00D4 --      3616                  SYNCHRONIZE; NOTSKIP:=FALSE
00D8 -1      3617               END
00E4 -0 A    3618      END; (* OF SETELEMLST *)
0124 --      3619
0124 -- A    3620      PROCEDURE SETRANGE;
0040 --      3621      (* <SET_RANGE> *)
0000 0- A    3622      BEGIN
000A --      3629         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3630         IF NOTSKIP THEN
001A --      3631         IF TOKENNR IN (.RBRACKSYM,LPARASYM,LBRACKSYM,NILSYM,NOTSYM,
001A --      3632                        STRINGSYM,PLUSMINUS,IDENTIFIER,UNSGREAL,UNSGINTEG.)
001A 1-      3633            THEN BEGIN
0054 2-      3634                    CASE TOKENNR OF
0062 --      3635                       RBRACKSYM:
0062 --      3636                          (* EPSILON PRODUCTION *);
0066 --      3637                       LPARASYM,LBRACKSYM,NILSYM,NOTSYM,STRINGSYM,PLUSMINUS,
0066 --      3638                       IDENTIFIER,UNSGREAL,UNSGINTEG:
0066 3-      3639                          BEGIN
0066 --      3640                             SETELEMENT;
006A --      3641                             SETELEMLST
006A -3      3642                          END
006E -2      3643                    END (* OF CASE *)
0138 -1      3644               END
0138 1-      3645            ELSE BEGIN
013C --      3646                    ERRMSG26(TOKENNR,'SETRANGE         ');
0166 --      3647                    SYNCHRONIZE; NOTSKIP:=FALSE
016A -1      3648               END
0176 -0 A    3649      END; (* OF SETRANGE *)
01B0 --      3650
01B0 -- A    3651      PROCEDURE FACTOR(VAR FACCOD: CHARVAR; VAR FACTYP: @TYPES);
0048 --      3652      (* <FACTOR> *)
0048 --      3653      VAR ID: HASHPTR;
004C --      3654          WORD: CHARVAR; (* WILL BE '$NOT', 'CHR' *)
0000 0- A    3655      BEGIN
000A --      3662         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      3663         IF NOTSKIP THEN
001A --      3664         IF TOKENNR IN (.NOTSYM,IDENTIFIER,STRINGSYM,UNSGINTEG,UNSGREAL,
001A --      3665                        NILSYM,LBRACKSYM,LPARASYM.)
001A 1-      3666            THEN BEGIN
0054 2-      3667                    CASE TOKENNR OF
0062 --      3668                       NOTSYM:
0062 3-      3669                          BEGIN TOPBOOL@.SWVAR:=FALSE;
0082 --      3670                             ADVANCE;
0086 --      3671                             FACTOR(FACCOD, FACTYP);
009A --      3672                             IF NOTSKIP AND (FACTYP@.TYPKLASS<>UNDEF) AND
00CE --      3673                                (FACTYP<>TYPBOOLEAN) THEN
00F2 --      3674                                ERRMSG(135,
00FA --      3675                                       'TYPE OF OPERAND MUST BE BOOLEAN         ');
0104 --      3676                             FACTYP:=TYPBOOLEAN;
012C --      3677                             LOC(WORD,5); LIT(WORD,5,'$NOT 6789012345');
015A --      3678                             CAT(FACCOD, WORD, FACCOD); FREE(WORD);
```

```
01C8 --      3679                            WRAP('(',FACCOD,')')
01E0 -3      3680                         END;
01E8 --      3681                      IDENTIFIER:
01E8 3-      3682                         BEGIN ID:=TOKENFD;
0208 --      3683                            ADVANCE;
020C --      3684                            IDENTREM(ID, FACCOD, FACTYP);
023A -3      3685                         END;
023E --      3686                      STRINGSYM:
023E 3-      3687                         BEGIN TOPBOOL@.SWVAR:=FALSE;
025E --      3688                            IF TOKENFD MOD MAXSTRGL=1
0270 4-      3689                            THEN BEGIN
0278 --      3690                                    FACTYP:=TYPCHAR;
02A0 --      3691                                    CODINTEG(ORD(STRING(.TOKENFD DIV MAXSTRGL.)),
02D0 --      3692                                             FACCOD);
02DC --      3693                                    LOC(WORD,4); LIT(WORD,4,'CHR 56789012345');
030A --      3694                                    CAT(FACCOD, WORD, FACCOD); FREE(WORD);
0378 --      3695                                    WRAP('(',FACCOD,')')
0390 -4      3696                                 END
0394 --      3697                            ELSE FACTYP:=BLDSTRG(TOKENFD);
03CC --      3698                            ADVANCE
03CC -3      3699                         END;
03D4 --      3700                      UNSGINTEG:
03D4 3-      3701                         BEGIN
03D4 --      3702                            TOPBOOL@.SWVAR:=FALSE;
03F4 --      3703                            FACTYP:=TYPINTEGER;
041C --      3704                            CODINTEG(TOKENFD, FACCOD);
0436 --      3705                            ADVANCE;
043A -3      3706                         END;
043E --      3707                      UNSGREAL:
043E 3-      3708                         BEGIN TOPBOOL@.SWVAR:=FALSE; ADVANCE;
0462 -3      3709                            FACTYP:=SRCHTAB(.SPREAL.)@.ACTLSTPTR@.TYP END;
04D6 --      3710                      NILSYM:
04D6 3-      3711                         BEGIN
04D6 --      3712                            TOPBOOL@.SWVAR:=FALSE;
04F6 --      3713                            ADVANCE;
04FA --      3714                            FACTYP:=NILTYP;
0522 --      3715                            LIT(FACCOD,5,'#NIL 6789012345')
0538 -3      3716                         END;
0540 --      3717                      LBRACKSYM:
0540 3-      3718                         BEGIN TOPBOOL@.SWVAR:=FALSE;
0560 --      3719                            ERRMSG(398,
0568 --      3720                                   'SETS ARE NOT SUPPORTED                  ');
0572 --      3721                            ADVANCE;
0576 --      3722                            SETRANGE; FACTYP:=UNDFZERO;
05A2 --      3723                            TERMINAL(RBRACKSYM,DUMMYFD)
05B4 -3      3724                         END;
05BC --      3725                      LPARASYM:
05BC 3-      3726                         BEGIN
05BC --      3727                            ADVANCE;
05C0 --      3728                            EXPRESSION(FACCOD, FACTYP);
05D4 --      3729                            IF FACTYP@.TYPKLASS<>UNDEF THEN
05F8 --      3730                               TOPBOOL@.SWVAR:=TOPBOOL@.UP@.SWVAR;
065A --      3731                            TERMINAL(RPARASYM,DUMMYFD)
066C -3      3732                         END
0670 -2      3733                   END (* OF CASE *)
0736 -1      3734                END
0736 1-      3735             ELSE BEGIN
073A --      3736                   ERRMSG26(TOKENNR,'FACTOR           ');
```

```
0764 --      3737                    SYNCHRONIZE; NOTSKIP:=FALSE; FACTYP:=UNDFZERO
0774 -1      3738                END
079C --      3769             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
079C -0 A    3770          END; (* OF FACTOR *)
086C --      3771
086C -- A    3772          PROCEDURE FACTORLIST(PRIORCOD: CHARVAR; PRIORTYP: @TYPES;
0040 --      3773                               VAR RETCOD: CHARVAR; VAR RETTYP: @TYPES);
0050 --      3774          (* <FACTOR_LIST> *)
0050 --      3775          VAR CUROPCOD, LEFTOPCOD: CHARVAR;
0058 --      3776              CUROPTYP, LEFTOPTYP: @TYPES; SHOULDBE: INTEGER;
0064 --      3777          (* SHOULDBE FLAGS OPERATION: 1: INTEGER, 2: BOOLEAN *)
0000 0- A    3778          BEGIN
000A --      3779             LOC(CUROPCOD, 50); LOC(LEFTOPCOD, 50);
0032 --      3786             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0032 --      3787             IF NOTSKIP THEN
0042 --      3788             IF TOKENNR IN (.MULTOPER,RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,
0042 --      3789                            RBRACKSYM,DOUBLEDOT,DOSYM,OFSYM,ORSYM,TOSYM,ENDSYM,
0042 --      3790                            ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM,PLUSMINUS,RELOPER.)
0042 1-      3791                THEN BEGIN
007C 2-      3792                        CASE TOKENNR OF
008A --      3793                           MULTOPER:
008A 3-      3794                              BEGIN
008A --      3795                                 TOPBOOL@.SWVAR:=FALSE;
00AA 4-      3796                                 CASE TOKENFD OF
00B8 5-      3797                                 1: BEGIN SHOULDBE:=1;
00C0 -5      3798                                       LIT(LEFTOPCOD,6,'$MULT 789012345') END;
00DE 5-      3799                                 2: BEGIN SHOULDBE:=1;
00E6 -5      3800                                       LIT(LEFTOPCOD,9,'#REALDIV 012345') END;
0104 5-      3801                                 3: BEGIN SHOULDBE:=1;
010C -5      3802                                       LIT(LEFTOPCOD,5,'$DIV 6789012345') END;
012A 5-      3803                                 4: BEGIN SHOULDBE:=1;
0132 -5      3804                                       LIT(LEFTOPCOD,5,'$MOD 6789012345') END;
0150 5-      3805                                 5: BEGIN SHOULDBE:=2;
0158 -5      3806                                       LIT(LEFTOPCOD,5,'$AND 6789012345') END
0172 -4      3807                                 END; (*CASE*)
01A4 --      3808                                 APP(LEFTOPCOD, PRIORCOD);
01CE --      3809                                 WRAP('(',LEFTOPCOD,')');
01EA --      3810                                 ADVANCE;
01EE --      3811                                 FACTOR(CUROPCOD, CUROPTYP);
0202 --      3812                                 IF COMPOPNDS(PRIORTYP, CUROPTYP, SHOULDBE)
0246 --      3813                                    THEN LEFTOPTYP:=CUROPTYP
0254 --      3814                                    ELSE LEFTOPTYP:=UNDFZERO;
029A --      3815                                 APP(LEFTOPCOD, CUROPCOD); WRAP('(',LEFTOPCOD,')');
02E0 --      3816                                 FACTORLIST(LEFTOPCOD, LEFTOPTYP, RETCOD, RETTYP)
032C -3      3817                              END;
0334 --      3818                           RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,RBRACKSYM,DOUBLEDOT,
0334 --      3819                           DOSYM,OFSYM,ORSYM,TOSYM,ENDSYM,ELSESYM,THENSYM,UNTILSYM,
0334 --      3820                           DOWNTOSYM,PLUSMINUS,RELOPER:
0334 --      3821                              (* EPSILON PRODUCTION *)
0334 3-      3822                              BEGIN
0334 --      3823                                 RETTYP:=PRIORTYP; SSS(RETCOD, PRIORCOD)
037C -3      3824                              END
0380 -2      3825                        END (* OF CASE *)
0482 -1      3826                    END
0482 1-      3827                ELSE BEGIN
0486 --      3828                        ERRMSG26(TOKENNR,'FACTORLIST       ');
04B0 --      3829                        SYNCHRONIZE; NOTSKIP:=FALSE; RETTYP:=UNDFZERO
04C0 -1      3830                    END
```

```
04E8 --      3831          ELSE RETTYP:=UNDFZERO; (* ALSO IN ERRORMODE *)
0514 --      3832          FREE(CUROPCOD); FREE(LEFTOPCOD)
0554 -0 A    3833       END; (* OF FACTORLIST *)
05E4 --      3834
05E4 -- A    3835       PROCEDURE TERM(VAR TERMCOD: CHARVAR; VAR TERMTYP: @TYPES);
0048 --      3836       (* <TERM> *)
0048 --      3837       VAR CUROPCOD: CHARVAR; CUROPTYP: @TYPES;
0000 0- A    3838       BEGIN
000A --      3839          LOC(CUROPCOD, 50);
001E --      3846          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --      3847          IF NOTSKIP THEN
002E --      3848          IF TOKENNR IN (.LPARASYM,LBRACKSYM,NILSYM,NOTSYM,STRINGSYM,
002E --      3849                         IDENTIFIER,UNSGREAL,UNSGINTEG.)
002E 1-      3850             THEN BEGIN
0068 --      3851                     FACTOR(CUROPCOD, CUROPTYP);
007C --      3852                     FACTORLIST(CUROPCOD, CUROPTYP, TERMCOD, TERMTYP)
00C8 -1      3853                  END
00CC 1-      3854             ELSE BEGIN
00D0 --      3855                     ERRMSG26(TOKENNR,'TERM             ');
00FA --      3856                     SYNCHRONIZE; NOTSKIP:=FALSE; TERMTYP:=UNDFZERO
010A -1      3857                  END;
0132 --      3888          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0132 --      3889          FREE(CUROPCOD)
0150 -0 A    3890       END; (* OF TERM *)
0188 --      3891
0188 -- A    3892       PROCEDURE TERMLIST(PRIORCOD: CHARVAR; PRIORTYP: @TYPES;
0040 --      3893                          VAR RETCOD: CHARVAR; VAR RETTYP: @TYPES);
0050 --      3894       (* <TERM_LIST> *)
0050 --      3895       VAR CUROPCOD, LEFTOPCOD: CHARVAR; CUROPTYP, LEFTOPTYP: @TYPES;
0000 0- A    3896       BEGIN
000A --      3897          LOC(CUROPCOD, 50); LOC(LEFTOPCOD, 50);
0032 --      3904          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0032 --      3905          IF NOTSKIP THEN
0042 --      3906          IF TOKENNR IN (.PLUSMINUS,ORSYM,RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,
0042 --      3907                         RBRACKSYM,DOUBLEDOT,DOSYM,OFSYM,TOSYM,ENDSYM,
0042 --      3908                         ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM,RELOPER.)
0042 1-      3909             THEN BEGIN
007C 2-      3910                     CASE TOKENNR OF
008A --      3911                        PLUSMINUS:
008A 3-      3912                           BEGIN
008A --      3913                              TOPBOOL@.SWVAR:=FALSE;
00AA --      3914                              IF TOKENFD=1
00AA --      3915                                 THEN LIT(LEFTOPCOD,6,'$PLUS 789012345')
00D2 --      3916                                 ELSE LIT(LEFTOPCOD,7,'$MINUS 89012345');
00F4 --      3917                              APP(LEFTOPCOD, PRIORCOD);
011E --      3918                              WRAP('(',LEFTOPCOD,')');
013A --      3919                              ADVANCE;
013E --      3920                              TERM(CUROPCOD, CUROPTYP);
0152 --      3921                              IF COMPOPNDS(PRIORTYP, CUROPTYP, 1)
0196 --      3922                                 THEN LEFTOPTYP:=CUROPTYP
01A4 --      3923                                 ELSE LEFTOPTYP:=UNDFZERO;
01EA --      3924                              APP(LEFTOPCOD, CUROPCOD); WRAP('(',LEFTOPCOD,')');
0230 --      3925                              TERMLIST(LEFTOPCOD, LEFTOPTYP, RETCOD, RETTYP)
027C -3      3926                           END;
0284 --      3927                        ORSYM:
0284 3-      3928                           BEGIN
0284 --      3929                              TOPBOOL@.SWVAR:=FALSE;
02A4 --      3930                              LIT(LEFTOPCOD,4,'$OR 56789012345');
```

```
02BE --      3931                              APP(LEFTOPCOD, PRIORCOD);
02E8 --      3932                              WRAP('(',LEFTOPCOD,')');
0304 --      3933                              ADVANCE;
0308 --      3934                              TERM(CUROPCOD, CUROPTYP);
031C --      3935                              IF COMPOPNDS(PRIORTYP, CUROPTYP, 2)
0360 --      3936                                 THEN LEFTOPTYP:=CUROPTYP
036E --      3937                                 ELSE LEFTOPTYP:=UNDFZERO;
03B4 --      3938                              APP(LEFTOPCOD, CUROPCOD); WRAP('(',LEFTOPCOD,')');
03FA --      3939                              TERMLIST(LEFTOPCOD, LEFTOPTYP, RETCOD, RETTYP)
0446 -3      3940                           END;
044E --      3941                        RPARASYM,SEMICSYM,COMMASYM,EQUALSYM,RBRACKSYM,DOUBLEDOT,
044E --      3942                        DOSYM,OFSYM,TOSYM,ENDSYM,ELSESYM,THENSYM,UNTILSYM,
044E --      3943                        DOWNTOSYM,RELOPER:
044E --      3944                           (* EPSILON PRODUCTION *)
044E 3-      3945                           BEGIN
044E --      3946                              RETTYP:=PRIORTYP; SSS(RETCOD, PRIORCOD)
0496 -3      3947                           END
049A -2      3948                     END (* OF CASE *)
059C -1      3949                  END
059C 1-      3950               ELSE BEGIN
05A0 --      3951                     ERRMSG26(TOKENNR,'TERMLIST         ');
05CA --      3952                     SYNCHRONIZE; NOTSKIP:=FALSE; RETTYP:=UNDFZERO
05DA -1      3953                  END
0602 --      3954            ELSE RETTYP:=UNDFZERO;
062E --      3955            FREE(CUROPCOD); FREE(LEFTOPCOD)
066E -0 A    3956         END; (* OF TERMLIST *)
06DC --      3957
06DC -- A    3958         PROCEDURE SIMPLEEXPR(VAR SEXPCOD: CHARVAR; VAR SEXPTYP: @TYPES);
0048 --      3959         (* <SIMPLE_EXPRESSION> *)
0048 --      3960         VAR CUROPCOD: CHARVAR; CUROPTYP: @TYPES;
0000 0- A    3961         BEGIN
000A --      3962            LOC(CUROPCOD, 50);
001E --      3969            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --      3970            IF NOTSKIP THEN
002E --      3971            IF TOKENNR IN (.PLUSMINUS,LPARASYM,LBRACKSYM,NILSYM,NOTSYM,
002E --      3972                           STRINGSYM,IDENTIFIER,UNSGREAL,UNSGINTEG.)
002E 1-      3973               THEN BEGIN
0068 2-      3974                     CASE TOKENNR OF
0076 --      3975                        PLUSMINUS:
0076 3-      3976                           BEGIN TOPBOOL@.SWVAR:=FALSE;
0096 --      3977                              IF TOKENFD=1
0096 --      3978                                 THEN LIT(SEXPCOD, 4,'$ID 56789012345')
00BE --      3979                                 ELSE LIT(SEXPCOD,12,'$MINUSUNARY 345');
00E0 --      3980                              ADVANCE;
00E4 --      3981                              TERM(CUROPCOD, CUROPTYP);
00F8 --      3982                              IF COMPOPNDS(CUROPTYP, CUROPTYP, 1)
013C --      3983                                 THEN SEXPTYP:=CUROPTYP
014A --      3984                                 ELSE SEXPTYP:=UNDFZERO;
0198 --      3985                              APP(SEXPCOD,CUROPCOD); WRAP('(',SEXPCOD,')')
01DA -3      3986                           END;
01E2 --      3987                        LPARASYM,LBRACKSYM,NILSYM,NOTSYM,STRINGSYM,IDENTIFIER,
01E2 --      3988                        UNSGREAL,UNSGINTEG:
01E2 3-      3989                           BEGIN
01E2 --      3990                              TERM(CUROPCOD, CUROPTYP);
01F6 --      3991                              TERMLIST(CUROPCOD, CUROPTYP, SEXPCOD, SEXPTYP)
0242 -3      3992                           END
0246 -2      3993                     END (* OF CASE *)
0310 -1      3994                  END
```

```
0310 1-      3995            ELSE BEGIN
0314 --      3996                    ERRMSG26(TOKENNR,'SIMPLEEXPRESSION ');
033E --      3997                    SYNCHRONIZE; NOTSKIP:=FALSE
0342 -1      3998                END;
034E --      4029          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
034E --      4030          FREE(CUROPCOD)
036C -0 A    4031       END; (* OF SIMPLEEXPR *)
03D0 --      4032
03D0 -- A    4033       PROCEDURE SIMPEXPREM(PRIORCOD: CHARVAR; PRIORTYP: @TYPES;
0040 --      4034                            VAR RETCOD: CHARVAR; VAR RETTYP: @TYPES);
0050 --      4035       (* <SIMPLE_EXPRESSION_REMAINDER> *)
0050 --      4036       VAR SECOPCOD: CHARVAR; SECOPTYP: @TYPES; SHOULDBE: INTEGER;
005C --      4037       (* SHOULBE FLAGS TYPE OF OPERATION: 3: =, <>, 4: ORDER, 5: IN *)
0000 0- A    4038       BEGIN
000A --      4039          LOC(SECOPCOD, 50);
001E --      4046          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --      4047          IF NOTSKIP THEN
002E --      4048          IF TOKENNR IN (.EQUALSYM,RELOPER,RPARASYM,SEMICSYM,COMMASYM,
002E --      4049                         RBRACKSYM,DOUBLEDOT,DOSYM,OFSYM,TOSYM,ENDSYM,
002E --      4050                         ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM.)
002E 1-      4051             THEN BEGIN
0068 2-      4052                     CASE TOKENNR OF
0076 --      4053                        EQUALSYM:
0076 3-      4054                           BEGIN
0076 --      4055                              TOPBOOL@.SWVAR:=FALSE;
0096 --      4056                              LIT(RETCOD,4,'$EQ 56789012345'); APP(RETCOD, PRIORCOD);
00DA --      4057                              WRAP('(',RETCOD,')');
00F6 --      4058                              ADVANCE;
00FA --      4059                              SIMPLEEXPR(SECOPCOD, SECOPTYP);
010E --      4060                              IF COMPOPNDS(PRIORTYP, SECOPTYP, 3)
0152 --      4061                                THEN RETTYP:=TYPBOOLEAN
0160 --      4062                                ELSE RETTYP:=UNDFZERO;
01B4 --      4063                              APP(RETCOD, SECOPCOD); WRAP('(', RETCOD, ')')
01F6 -3      4064                           END;
01FE --      4065                        RELOPER:
01FE 3-      4066                           BEGIN
01FE --      4067                              TOPBOOL@.SWVAR:=FALSE;
021E 4-      4068                              CASE TOKENFD OF
022C 5-      4069                              2: BEGIN SHOULDBE:=3;
0234 -5      4070                                    LIT(RETCOD,4,'$NE 56789012345') END;
0252 5-      4071                              3: BEGIN SHOULDBE:=4;
025A -5      4072                                    LIT(RETCOD,4,'$LT 56789012345') END;
0278 5-      4073                              4: BEGIN SHOULDBE:=4;
0280 -5      4074                                    LIT(RETCOD,4,'$GT 56789012345') END;
029E 5-      4075                              5: BEGIN SHOULDBE:=4;
02A6 -5      4076                                    LIT(RETCOD,4,'$LE 56789012345') END;
02C4 5-      4077                              6: BEGIN SHOULDBE:=4;
02CC -5      4078                                    LIT(RETCOD,4,'$GE 56789012345') END;
02EA 5-      4079                              7: BEGIN SHOULDBE:=5;
02F2 -5      4080                                    LIT(RETCOD,4,'#IN 56789012345') END
030C -4      4081                              END; (*CASE*)
0342 --      4082                              APP(RETCOD, PRIORCOD);
036C --      4083                              WRAP('(',RETCOD,')');
0388 --      4084                              ADVANCE;
038C --      4085                              SIMPLEEXPR(SECOPCOD, SECOPTYP);
03A0 --      4086                              IF COMPOPNDS(PRIORTYP, SECOPTYP, SHOULDBE)
03E4 --      4087                                THEN RETTYP:=TYPBOOLEAN
03F2 --      4088                                ELSE RETTYP:=UNDFZERO;
```

```
0446 --      4089                              APP(RETCOD, SECOPCOD); WRAP('(',RETCOD,')')
0488 -3      4090                           END;
0490 --      4091                        RPARASYM,SEMICSYM,COMMASYM,RBRACKSYM,DOUBLEDOT,DOSYM,
0490 --      4092                        OFSYM,TOSYM,ENDSYM,ELSESYM,THENSYM,UNTILSYM,DOWNTOSYM:
0490 --      4093                           (* EPSILON PRODUCTION *)
0490 3-      4094                           BEGIN
0490 --      4095                              RETTYP:=PRIORTYP; SSS(RETCOD, PRIORCOD)
04D8 -3      4096                           END
04DC -2      4097                     END (* OF CASE *)
05DE -1      4098                  END
05DE 1-      4099               ELSE BEGIN
05E2 --      4100                     ERRMSG26(TOKENNR,'SIMPLEEXPRESSREM ');
060C --      4101                     SYNCHRONIZE; NOTSKIP:=FALSE; RETTYP:=UNDFZERO
061C -1      4102                  END
0644 --      4103            ELSE RETTYP:=UNDFZERO;
0670 --      4104            FREE(SECOPCOD)
068E -0 A    4105         END; (* OF SIMPEXPREM *)
073C --      4106
073C -- A    4107         PROCEDURE EXPRESSION (*VAR EXPCOD: CHARVAR; VAR EXPTYP: @TYPES*);
0048 --      4108         (* <EXPRESSION> *)
0048 --      4109         VAR CUROPCOD: CHARVAR; CUROPTYP: @TYPES;
0000 0- A    4110         BEGIN
000A --      4111            LOC(CUROPCOD, 50);
001E --      4118            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --      4119            IF NOTSKIP THEN
002E --      4120            IF TOKENNR IN (.LPARASYM,LBRACKSYM,NILSYM,NOTSYM,STRINGSYM,
002E --      4121                           PLUSMINUS,IDENTIFIER,UNSGREAL,UNSGINTEG.)
002E 1-      4122               THEN BEGIN
0068 --      4123                     (* STACK ONE BOOLEAN WITH SWVAR TRUE *)
0068 2-      4124                     IF TOPBOOL@.UP=NIL THEN BEGIN
008E --      4125                        NEW(TOPBOOL@.UP);
00B8 --      4126                        TOPBOOL@.UP@.DOWN:=TOPBOOL;
010A -2      4127                        TOPBOOL@.UP@.UP:=NIL END;
013E --      4128                     TOPBOOL:=TOPBOOL@.UP; TOPBOOL@.SWVAR:=TRUE;
019E --      4129                     SIMPLEEXPR(CUROPCOD, CUROPTYP);
01B2 --      4130                     SIMPEXPREM(CUROPCOD, CUROPTYP, EXPCOD, EXPTYP);
0202 --      4131                     (* POP ONE BOOLEAN *)
0202 --      4132                     TOPBOOL:=TOPBOOL@.DOWN
021C -1      4133                  END
0240 1-      4134               ELSE BEGIN
0244 --      4135                     ERRMSG26(TOKENNR,'EXPRESSION       ');
026E --      4136                     SYNCHRONIZE; NOTSKIP:=FALSE; EXPTYP:=UNDFZERO
027E -1      4137                  END
02A6 --      4138            ELSE EXPTYP:=UNDFZERO;
02D2 --      4169            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
02D2 --      4170            FREE(CUROPCOD)
02F0 -0 A    4171         END; (* OF EXPRESSION *)
032C --      4172
032C -- A    4173         PROCEDURE SIMPSTMREM(ID: HASHPTR);
0044 --      4174         (* <SIMPLE_STATEMENT_REMAINDER> *)
0044 --      4175         LABEL 1;
0044 --      4176         VAR PTRACT: @ACTIVS; TYPIN, TYPOUT, EXPTYP: @TYPES;
0054 --      4177             STACKTRAV: @ACTIVS;
0058 --      4178             REPLLST, TRAVRL, SECLAST: @SUBSBNDS;
0064 --      4179             EXPCOD, STMCOD, NAMELST, REPLMENT, SLICES, HELPER: CHARVAR;
007C --      4180             ACTEXPLST: @EXPLST; RUNACT: @EXPLST; OLDARG, RUNARG, ARGLST: @ARGS;
0000 0- A    4181         BEGIN
000A --      4182            LOC(EXPCOD, 50); LOC(REPLMENT, 50); LOC(SLICES, 50); LOC(HELPER, 50);
```

```
005A --    4183          LOC(NAMELST, 150); LOC(STMCOD, 200);
0082 --    4190          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
0082 --    4191          IF NOTSKIP THEN
0092 --    4192          IF TOKENNR IN (.PERIODSYM,LBRACKSYM,POINTER,BECOMES,LPARASYM,
0092 --    4193                         SEMICSYM,ENDSYM,UNTILSYM,ELSESYM.)
0092 1-    4194             THEN BEGIN
00CC 2-    4195                     CASE TOKENNR OF
00DA --    4196                        PERIODSYM,LBRACKSYM,POINTER,BECOMES:
00DA 3-    4197                           BEGIN
00DA --    4198                              PTRACT:=SEARCHALL(ID, VARID);
011E 4-    4199                              CASE PTRACT@.IDKLASS OF
013A 5-    4200                              SCALID,TYPID,PROCID: BEGIN ERRMSG103(ID);
0158 -5    4201                                                   TYPIN:=UNDFZERO END;
0180 --    4202                              FLDID,VARID: TYPIN:=PTRACT@.TYP;
01B6 --    4203                              FUNCID: IF (PTRACT@.LEVNR<>0) AND
01DC --    4204                                         (PTRACT@.LEVNR<LEVEL)
0208 --    4205                                         THEN TYPIN:=PTRACT@.RETTYP
0224 5-    4206                                         ELSE BEGIN ERRMSG(151,
024E --    4207                                           'ASSIGNMENT TO FUNCTION OUT OF ITS BODY  ');
0258 -5    4208                                                TYPIN:=UNDFZERO END;
0280 -4    4209                              END; (*CASE*)
02B2 --    4210                              REPLLST:=NIL;
02B8 --    4211                              VARSELECTS(TRUE, REPLLST,
02C8 --    4212                                         DUMMYCOD, TYPIN, DUMMYCOD, TYPOUT);
0320 --    4213                              TERMINAL(BECOMES,DUMMYFD);
0336 --    4214                              EXPRESSION(EXPCOD,EXPTYP);
034A --    4215                              COMPARE(TYPOUT, EXPTYP);
038A --    4216                              (* GENERATE THE FULL CODE FOR THIS STATEMENT *)
038A --    4217                              (* OBTAIN CORRECT REPLACEMENT *)
038A --    4218                              IF REPLLST=NIL
038A --    4219                                 THEN SSS(REPLMENT, EXPCOD)
03BC 4-    4220                                 ELSE BEGIN (* MUST BE A STRUCTURE *)
03C4 --    4221                                      SSS(REPLMENT, EXPCOD);
03EE 5-    4222                                      REPEAT
03EE --    4223                                         TRAVRL:=REPLLST; SECLAST:=NIL;
0412 --    4224                                         CODIDENT(PTRACT, SLICES);
043C --    4225                                         WHILE TRAVRL@.NXTSBND<>NIL DO
045C 6-    4226                                            BEGIN
045C --    4227                                            SSS(HELPER, RETRIEVE);
048C --    4228                                            APP(HELPER, TRAVRL@.SUB);
04CA --    4229                                            WRAP('(',HELPER,')');
04E6 --    4230                                            APP(HELPER, TRAVRL@.BND);
0524 --    4231                                            WRAP('(',HELPER,')');
0540 --    4232                                            APP(SLICES, HELPER);
056A --    4233                                            WRAP('(',SLICES,')');
0586 --    4234                                            SECLAST:=TRAVRL; TRAVRL:=TRAVRL@.NXTSBND;
05D6 -6    4235                                            END;
05DA --    4236                                         SSS(HELPER, REPLACE);
060A --    4237                                         APP(HELPER, TRAVRL@.SUB);
0648 --    4238                                         WRAP('(',HELPER,')');
0664 --    4239                                         APP(HELPER, TRAVRL@.BND);
06A2 --    4240                                         WRAP('(',HELPER,')');
06BE --    4241                                         APP(HELPER, REPLMENT);
06E8 --    4242                                         WRAP('(',HELPER,')');
0704 --    4243                                         APP(SLICES, HELPER);
072E --    4244                                         WRAP('(',SLICES,')');
074A --    4245                                         (* CUT OFF LAST LIST ELEMENT *)
074A --    4246                                         FREE(TRAVRL@.SUB); FREE(TRAVRL@.BND);
```

```
07B6 --      4247                                    IF SECLAST<>NIL THEN
07C2 --      4248                                       SECLAST@.NXTSBND:=NIL;
07DC --      4249                                    (* EXCHANGE VARIABLES *)
07DC --      4250                                    SSS(REPLMENT, SLICES)
0802 -5      4251                                 UNTIL SECLAST=NIL
0806 -4      4252                                 END; (* REPLMENT HOLDS SUBSTITUTE *)
0812 --      4253                              (* PUZZLE STATEMENT TOGETHER *)
0812 --      4254                              ENVIRON(STMCOD);
081E --      4255                              LIT(NAMELST, 5, '$PHI 6789012345');
0838 --      4256                              STACKTRAV:=STACK;
085C --      4257                              WHILE STACKTRAV<>NIL DO
0868 4-      4258                                 BEGIN
0868 --      4259                                    IF STACKTRAV=PTRACT
0868 5-      4260                                       THEN BEGIN
0874 --      4261                                            APP(NAMELST, REPLMENT);
089E --      4262                                            WRAP('(',NAMELST,')')
08B6 -5      4263                                            END
08BA 5-      4264                                       ELSE CASE STACKTRAV@.IDKLASS OF
08DA --      4265                                            SCALID, TYPID, PROCID:;
08DE --      4266                                            VARID, FLDID:
08DE 6-      4267                                               BEGIN
08DE --      4268                                               CODIDENT(STACKTRAV, HELPER);
0908 --      4269                                               APP(NAMELST, HELPER);
0932 --      4270                                               WRAP('(',NAMELST,')')
094A -6      4271                                               END;
0952 --      4272                                            FUNCID:
0952 --      4273                                               IF (STACKTRAV@.LEVNR<>0) AND
0978 --      4274                                                  (STACKTRAV@.LEVNR<LEVEL)
09A4 6-      4275                                                  THEN BEGIN
09AC --      4276                                                       CODIDENT(STACKTRAV, HELPER);
09D6 --      4277                                                       CAT(HELPER, PI, HELPER);
0A24 --      4278                                                       APP(NAMELST, HELPER);
0A4E --      4279                                                       WRAP('(',NAMELST,')')
0A66 -6      4280                                                       END
0A6A -5      4281                                            END;(*CASE*)
0AA0 --      4282                                    STACKTRAV:=STACKTRAV@.NXTACT
0AB4 -4      4283                                 END;
0AD6 --      4284                              APP(STMCOD, NAMELST);
0B00 --      4285                              WRAP('(',STMCOD,')');
0B1C --      4286                              STMOUT(STMNR, STMCOD);
0B4C -3      4287                           END;
0B50 --      4288                        LPARASYM,SEMICSYM,ENDSYM,UNTILSYM,ELSESYM:
0B50 3-      4289                           BEGIN
0B50 --      4290                              IF (ID=SPREAD) OR (ID=SPWRITE) OR
0B82 --      4291                                 (ID=SPREADLN) OR (ID=SPWRITELN)
0BB4 4-      4292                                 THEN BEGIN
0BBC --      4293                                         ERRMSG(398,
0BC4 --      4294                                           'ONLY GET/PUT SUPPORTED                  ');
0BCE --      4295                                         IF TOKENNR=LPARASYM
0BCE --      4296                                           THEN ACTPARAMPT(ACTEXPLST);
0BEC --      4297                                      GOTO 1
0BF0 -4      4298                                      END;
0BF0 --      4299                              IF (ID=SPGET) OR (ID=SPPUT)
0C20 4-      4300                                 THEN BEGIN IF TOKENNR=LPARASYM
0C28 5-      4301                                               THEN BEGIN ERRMSG(398,
0C42 --      4302                                                 'PARAMETER FOR GET/PUT IGNORED           ');
0C4C --      4303                                                          ACTPARAMPT(ACTEXPLST)
0C54 -5      4304                                                    END;
```

```
0C58 --      4305                (* CONSTRUCT CODE FOR BOTH *)
0C58 --      4306                ENVIRON(STMCOD);
0C64 --      4307                IF ID=SPGET THEN
0C76 5-      4308                   BEGIN
0C76 --      4309                   PTRACT:=SEARCHALL(SPINPUT, VARID);
0CC0 --      4310                   LIT(REPLMENT, 7, '$SCARDS89012345');
0CDA --      4311                   LIT(NAMELST, 5, '$PHI 6789012345');
0CF4 --      4312                   STACKTRAV:=STACK;
0D18 --      4313                   WHILE STACKTRAV<>NIL DO
0D24 6-      4314                      BEGIN
0D24 --      4315                      IF STACKTRAV=PTRACT
0D24 7-      4316                         THEN BEGIN
0D30 --      4317                              APP(NAMELST, REPLMENT);
0D5A --      4318                              WRAP('(',NAMELST,')')
0D72 -7      4319                              END
0D76 7-      4320                         ELSE CASE STACKTRAV@.IDKLASS OF
0D96 --      4321                              SCALID, TYPID, PROCID:;
0D9A --      4322                              VARID, FLDID:
0D9A 8-      4323                                 BEGIN
0D9A --      4324                                   CODIDENT(STACKTRAV, HELPER);
0DC4 --      4325                                   APP(NAMELST, HELPER);
0DEE --      4326                                   WRAP('(',NAMELST,')')
0E06 -8      4327                                 END;
0E0E --      4328                              FUNCID:
0E0E --      4329                                 IF (STACKTRAV@.LEVNR<>0) AND
0E34 --      4330                                 (STACKTRAV@.LEVNR<LEVEL)
0E60 --      4331                                 THEN
0E68 8-      4332                                 BEGIN
0E68 --      4333                                   CODIDENT(STACKTRAV, HELPER);
0E92 --      4334                                   CAT(HELPER, PI, HELPER);
0EE0 --      4335                                   APP(NAMELST, HELPER);
0F0A --      4336                                   WRAP('(',NAMELST,')')
0F22 -8      4337                                 END
0F26 -7      4338                           END;(*CASE*)
0F5C --      4339                      STACKTRAV:=STACKTRAV@.NXTACT
0F70 -6      4340                      END;
0F92 --      4341                   LIT(HELPER, 7, '$SPRINT89012345');
0FAC --      4342                   APP(NAMELST, HELPER);
0FD6 --      4343                   WRAP('(',NAMELST,')');
0FF2 --      4344                   LIT(HELPER, 10, '@ $SPRINT,12345');
100C --      4345                   LIT(REPLMENT, 8, '$SCARDS:9012345');
1026 --      4346                   APP(HELPER, REPLMENT);
1050 --      4347                   CAT(NAMELST, HELPER, NAMELST);
1098 --      4348                   WRAP('(',NAMELST,')');
10B4 --      4349                   APP(STMCOD, NAMELST);
10DE --      4350                   WRAP('(',STMCOD,')');
10FA --      4351                   STMOUT(STMNR, STMCOD)
1126 -5      4352                   END;
112A --      4353                IF ID=SPPUT THEN
113C 5-      4354                   BEGIN
113C --      4355                   PTRACT:=SEARCHALL(SPOUTPUT, VARID);
1186 --      4356                   NAMELST@.CURLG:=0;
11A0 --      4357                   APPENV(NAMELST);
11AC --      4358                   LIT(HELPER, 15, '(($CAT $SPRINT)');
11C6 --      4359                   LIT(REPLMENT, 7, 'OUTPUT)89012345');
11E0 --      4360                   APP(HELPER, REPLMENT);
120A --      4361                   APP(NAMELST, HELPER);
1234 --      4362                   WRAP('(',NAMELST,')');
```

```
1250 --      4363                                  LIT(HELPER, 10, '@ $SPRINT:12345');
126A --      4364                                  CAT(NAMELST, HELPER, NAMELST);
12B2 --      4365                                  WRAP('(',NAMELST,')');
12CE --      4366                                  APP(STMCOD, NAMELST);
12F8 --      4367                                  WRAP('(',STMCOD,')');
1314 --      4368                                  STMOUT(STMNR, STMCOD)
1340 -5      4369                                  END;
1344 --      4370                               GOTO 1
1348 -4      4371                               END;
1348 4-      4372                            IF (ID=SPNEW) THEN BEGIN
135A --      4373                               ACTPARAMPT(ACTEXPLST);
1366 --      4374                               IF ACTEXPLST=NIL
1366 --      4375                                  THEN ERRMSG(126,
137A --      4376                                  'PROCEDURE ''NEW'' MUST HAVE PARAMETERS    ')
1380 --      4377                                  ELSE
1388 --      4378                                  IF NOT (ACTEXPLST@.EXPLSTTYP@.TYPKLASS IN
13B0 --      4379                                        (.PTRTYP, UNDEF.)) OR
13EA --      4380                                     NOT ACTEXPLST@.SWVAR
13FE --      4381                                     THEN ERRMSG(154,
1416 --      4382                                        'POINTER VARIABLE EXPECTED IN PROC. ''NEW''')
141C --      4383                                     ELSE IF ACTEXPLST@.NXTEXP<>NIL
1438 --      4384                                           THEN ERRMSG(398,
144C --      4385                                     'TAGFIELDVALUES IN PROC. ''NEW'' IGNORED   ');
1456 -4      4386                               GOTO 1 END;
145A --      4387                            PTRACT:=SEARCHALL(ID, PROCID);
149E --      4388                            IF PTRACT@.IDKLASS<>PROCID
14B2 44      4389                               THEN BEGIN ERRMSG103(ID); ARGLST:=NIL END
14E2 --      4390                               ELSE ARGLST:=PTRACT@.ARG;
1518 --      4391                            IF TOKENNR=LPARASYM
1518 --      4392                               THEN ACTPARAMPT(ACTEXPLST)
1532 --      4393                               (* EPSILON PRODUCTION *)
1532 --      4394                               ELSE ACTEXPLST:=NIL;
1540 --      4395                            IF PTRACT@.IDKLASS=PROCID THEN
1560 --      4396                               IF PTRACT@.RETTYP=NIL THEN
1580 4-      4397                                  BEGIN (* TAKE ACTUAL ONE *)
1580 --      4398                                     IF ACTEXPLST=NIL THEN ARGLST:=NIL
158C 5-      4399                                     ELSE BEGIN
1596 --      4400                                     RUNACT:=ACTEXPLST;
15B4 --      4401                                     WHILE RUNACT<>NIL DO
15C0 6-      4402                                        BEGIN NEW(RUNARG);
15D0 --      4403                                           IF ARGLST=NIL
15D0 --      4404                                              THEN ARGLST:=RUNARG
15DC --      4405                                              ELSE OLDARG@.NXTARG:=RUNARG;
1630 --      4406                                           RUNARG@.ARGTYP:=RUNACT@.EXPLSTTYP;
1676 --      4407                                           RUNARG@.PASSKND:=FALSE; OLDARG:=RUNARG;
16AE --      4408                                           RUNACT:=RUNACT@.NXTEXP
16C2 -5      4409                                        END; OLDARG@.NXTARG:=NIL END;
16FE --      4410                                        PTRACT@.RETTYP:=UNDFZERO;
1736 --      4411                                        PTRACT@.ARG:=ARGLST; PUSH(PTRACT);
178A -4      4412                                  END;
178A --      4413                            CODIDENT(PTRACT, NAMELST);
17B4 --      4414                            APP(NAMELST, LITBLANK);
17E4 --      4415                            WHILE (ACTEXPLST<>NIL) AND (ARGLST<>NIL) DO
1810 4-      4416                               BEGIN COMPARE(ACTEXPLST@.EXPLSTTYP,
1842 --      4417                                           ARGLST@.ARGTYP);
1878 --      4418                                  IF NOT ACTEXPLST@.SWVAR
188C --      4419                                     AND ARGLST@.PASSKND THEN
18B2 --      4420                                     ERRMSG(154,
```

```
18BA --    4421                                    'ACTUAL PARAMETER MUST BE A VARIABLE     ');
18C4 --    4422                                 (* TREAT PARAMETERS AS VALUE PARAMETERS *)
18C4 --    4423                                 APP(NAMELST, ACTEXPLST@.EXPLSTCOD);
1902 --    4424                                 FREE(ACTEXPLST@.EXPLSTCOD);
1938 --    4425                                 WRAP('(',NAMELST,')');
1954 --    4426                                 ACTEXPLST:=ACTEXPLST@.NXTEXP;
1986 -4    4427                                 ARGLST:=ARGLST@.NXTARG END;
19BC --    4428                           IF (ACTEXPLST<>NIL) OR (ARGLST<>NIL)
19E0 --    4429                              THEN ERRMSG(126,
19F0 --    4430                                    'ACTUAL NUMBER OF ARGUMENTS UNEQUALS DCL.');
19FA --    4431                           (* RESOLVE THE ENVIRONMENT CONFLICTS *)
19FA --    4432                           (* REPLMENT HOLDS COVERED INNER VARIABLES *)
19FA --    4433                           LIT(REPLMENT, 5, '$PHI 6789012345');
1A14 --    4434                           STACKTRAV:=STACK;
1A38 --    4435                           WHILE STACKTRAV<>NIL DO
1A44 4-    4436                              BEGIN CASE STACKTRAV@.IDKLASS OF
1A60 --    4437                                 SCALID, TYPID, PROCID, FUNCID:;
1A64 --    4438                                 VARID, FLDID:
1A64 --    4439                                    (* MUST VARIABLE BE COVERED? *)
1A64 --    4440                                    IF STACKTRAV@.LEVNR>PTRACT@.LEVNR
1A8C 6-    4441                                       THEN BEGIN
1A98 --    4442                                               CODIDENT(STACKTRAV, HELPER);
1AC2 --    4443                                               APP(REPLMENT, HELPER);
1AEC --    4444                                               WRAP('(',REPLMENT,')')
1B04 -6    4445                                            END
1B08 6-    4446                                       ELSE BEGIN
1B0C --    4447                                            (* REPLMENT APPENDED ALREADY? *)
1B0C --    4448                                            IF REPLMENT@.CURLG<>0 THEN
1B2C 7-    4449                                               BEGIN APP(NAMELST, REPLMENT);
1B56 --    4450                                                  WRAP('(',NAMELST,')');
1B72 -7    4451                                                  REPLMENT@.CURLG:=0 END;
1B8C --    4452                                            CODIDENT(STACKTRAV, HELPER);
1BB6 --    4453                                            APP(NAMELST, HELPER);
1BE0 --    4454                                            WRAP('(',NAMELST,')')
1BF8 -6    4455                                            END
1BFC -5    4456                                    END; (* CASE *)
1C38 --    4457                                    STACKTRAV:=STACKTRAV@.NXTACT
1C4C -4    4458                                 END; (*WHILE*)
1C6E --    4459                           ENVIRON(STMCOD);
1C7A --    4460                           APP(STMCOD, NAMELST);
1CA4 --    4461                           WRAP('(',STMCOD,')');
1CC0 --    4462                           STMOUT(STMNR, STMCOD);
1CF0 -3    4463                        1:END
1CF0 -2    4464                     END (* OF CASE *)
1DD8 -1    4465                  END
1DD8 1-    4466               ELSE BEGIN
1DDC --    4467                       ERRMSG26(TOKENNR,'SIMPLESTATEMNTREM');
1E06 --    4468                       SYNCHRONIZE; NOTSKIP:=FALSE
1E0A -1    4469                  END;
1E16 --    4470            FREE(EXPCOD); FREE(STMCOD); FREE(REPLMENT); FREE(SLICES); FREE(HELPER);
1EC0 --    4471            FREE(NAMELST);
1EE2 -0 A  4472         END; (* OF SIMPSTMREM *)
2114 --    4473
2114 -- A  4474         PROCEDURE COMPSTMREM(FIRSTSTMNR: INTEGER; VAR COMPSTMCOD: CHARVAR);
0048 --    4475         (* <COMPOUND_STATEMENT_REMAINDER> *)
0048 --    4476         VAR CURSTMNR: INTEGER; PREFIX: CHARVAR;
0000 0- A  4477         BEGIN
000A --    4478            LOC(PREFIX, 7);
```

```
001E --      4485          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
001E --      4486          IF TOKENNR IN (.SEMICSYM,ENDSYM.)
001E 1-      4487             THEN BEGIN
0058 2-      4488                     CASE TOKENNR OF
0066 --      4489                        SEMICSYM:
0066 3-      4490                           BEGIN
0066 --      4491                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0074 --      4492                              ADVANCE;
0078 --      4493                              CURSTMNR:=STMNR+1; (* TO SPARE PARAMETERS *)
008A --      4494                              STATEMENT;
008E --      4495                              COMPSTMREM(CURSTMNR, COMPSTMCOD)
009E -3      4496                           END;
00A6 --      4497                        ENDSYM:
00A6 3-      4498                           BEGIN
00A6 --      4499                              NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
00B4 --      4500                              ADVANCE; NEST:=NEST-1
00B8 -3      4501                           END
00D0 -2      4502                     END; (* OF CASE *)
0192 --      4503                     CODINTEG(FIRSTSTMNR, PREFIX);
01A6 --      4504                     CAT(PREFIX, STM, PREFIX);
01F4 --      4505                     CAT(COMPSTMCOD, PREFIX, COMPSTMCOD);
0240 --      4506                     WRAP('(',COMPSTMCOD,')')
0258 -1      4507                END
025C 1-      4508             ELSE BEGIN
0260 --      4509                     ERRMSG26(TOKENNR,'COMPOUNDSTMNTREM ');
028A --      4510                     GOTO 9999
0298 -1      4511                END;
0298 --      4512          FREE(PREFIX);
02BA -0 A    4513       END; (* OF COMPSTMREM *)
0300 --      4514
0300 -- A    4515       PROCEDURE ELSECLAUSE(VAR ELSESTMNR: INTEGER);
0044 --      4516       (* <ELSE_CLAUSE> *)
0044 --      4517       (* ELSESTMNR = -1 IF ELSE CLAUSE EMPTY *)
0000 0- A    4518       BEGIN
000A --      4525          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4526          IF NOTSKIP THEN
001A --      4527          IF TOKENNR IN (.ELSESYM,SEMICSYM,ENDSYM,UNTILSYM.)
001A 1-      4528             THEN BEGIN
0054 2-      4529                     CASE TOKENNR OF
0062 --      4530                        ELSESYM:
0062 3-      4531                           BEGIN
0062 --      4532                              ADVANCE;
0066 --      4533                              ELSESTMNR:=STMNR+1;
007C --      4534                              STATEMENT
007C -3      4535                           END;
0084 --      4536                        SEMICSYM,ENDSYM,UNTILSYM:
0084 --      4537                           ELSESTMNR:=-1
0084 --      4538                           (* EPSILON PRODUCTION *)
0084 -2      4539                     END (* OF CASE *)
0156 -1      4540                END
0156 1-      4541             ELSE BEGIN
015A --      4542                     ERRMSG26(TOKENNR,'ELSECLAUSE       ');
0184 --      4543                     SYNCHRONIZE; NOTSKIP:=FALSE
0188 -1      4544                END
0194 -0 A    4545       END; (* OF ELSECLAUSE *)
01D8 --      4546
01D8 -- A    4547       PROCEDURE CASEELEMNT(VAR CONSLST: @VALUS; MATCHTYP: @TYPES);
0048 --      4548       (* <CASE_ELEMENT> *)
```

```
0048 --      4549      VAR CONSTYP: @TYPES; CONSVAL: INTEGER;
0000 0- A    4550      BEGIN
000A --      4557         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4558         IF NOTSKIP THEN
001A --      4559         IF TOKENNR IN (.IDENTIFIER,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM,
001A --      4560                        SEMICSYM,ENDSYM.)
001A 1-      4561            THEN BEGIN
0054 2-      4562                    CASE TOKENNR OF
0062 --      4563                       IDENTIFIER,UNSGINTEG,PLUSMINUS,UNSGREAL,STRINGSYM:
0062 3-      4564                          BEGIN
0062 --      4565                             CONSTANT(CONSTYP, CONSVAL);
0076 --      4566                             CONSTLIST(CONSTYP,CONSVAL,CONSLST,MATCHTYP);
00C6 --      4567                             TERMINAL(COLONSYM,DUMMYFD);
00DC --      4568                             STATEMENT
00DC -3      4569                          END;
00E4 --      4570                       SEMICSYM,ENDSYM:
00E4 --      4571                          (* EPSILON PRODUCTION *)
00E4 --      4572                          CONSLST:=NIL
00E4 -2      4573                    END (* OF CASE *)
01DC -1      4574                END
01DC 1-      4575            ELSE BEGIN
01E0 --      4576                    ERRMSG26(TOKENNR,'CASEELEMENT      ');
020A --      4577                    SYNCHRONIZE; NOTSKIP:=FALSE
020E -1      4578                END
021A -0 A    4579      END; (* OF CASEELEMNT *)
0258 --      4580
0258 -- A    4581      PROCEDURE CASELEMLST(FIRCONSLST: @VALUS; VAR CARCONSLST: @VALUS;
0040 --      4582                           MATCHTYP: @TYPES);
004C --      4583      (* <CASE_ELEMENT_LIST> *)
004C --      4584      VAR CURCONSLST, CDRCONSLST: @VALUS; OLDCONS, NEWCONS: @VALUS; NOTFND: BOOLEAN;
0000 0- A    4585      BEGIN
000A --      4592         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4593         IF TOKENNR IN (.SEMICSYM,ENDSYM.)
000A 1-      4594            THEN BEGIN
0044 2-      4595                    CASE TOKENNR OF
0052 --      4596                       SEMICSYM:
0052 3-      4597                          BEGIN
0052 --      4598                             NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0060 --      4599                             ADVANCE;
0064 --      4600                             CASEELEMNT(CURCONSLST, MATCHTYP);
008E --      4601                             CASELEMLST(CURCONSLST, CDRCONSLST, MATCHTYP)
00D2 -3      4602                          END;
00DA --      4603                       ENDSYM:
00DA 3-      4604                          BEGIN
00DA --      4605                             NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
00E8 --      4606                             CDRCONSLST:=NIL;
00EE --      4607                             ADVANCE; NEST:=NEST-1
00F2 -3      4608                          END
010A -2      4609                    END; (* OF CASE *)
01CC 2-      4610                    WHILE FIRCONSLST<>NIL DO BEGIN
01D8 --      4611                       OLDCONS:=NIL; NEWCONS:=CDRCONSLST; NOTFND:=TRUE;
0204 --      4612                       WHILE (NEWCONS<>NIL) AND NOTFND DO
0220 --      4613                          IF NEWCONS@.VALEE<>FIRCONSLST@.VALEE
0248 3-      4614                             THEN BEGIN OLDCONS:=NEWCONS;
0272 -3      4615                                  NEWCONS:=NEWCONS@.NXTVAL END
02A4 --      4616                             ELSE NOTFND:=FALSE;
02B2 --      4617                       IF NOTFND
02B2 --      4618                          THEN IF OLDCONS<>NIL
```

```
02BC --      4619                                  THEN OLDCONS@.NXTVAL:=FIRCONSLST
02DC --      4620                                  ELSE CDRCONSLST:=FIRCONSLST
02FE --      4621                            ELSE ERRMSG(156,
0328 --      4622                                        'MULTIDEFINED CASE LABEL                ');
0332 --      4623                           OLDCONS:=FIRCONSLST;
0350 --      4624                           FIRCONSLST:=FIRCONSLST@.NXTVAL;
0382 -2      4625                           OLDCONS@.NXTVAL:=NIL END;
03A0 --      4626                           CARCONSLST:=CDRCONSLST
03A0 -1      4627                     END
03C2 1-      4628                ELSE BEGIN
03C6 --      4629                        ERRMSG26(TOKENNR,'CASEELEMENTLIST  ');
03F0 --      4630                        GOTO 9999
03FE -1      4631                     END
03FE -0 A    4632          END; (* OF CASELEMLST *)
0468 --      4633
0468 -- A    4634          PROCEDURE REPSTMLIST;
0040 --      4635          (* <REPEAT_STATEMENT_LIST> *)
0000 0- A    4636          BEGIN
000A --      4643             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4644             IF TOKENNR IN (.SEMICSYM,UNTILSYM.)
000A 1-      4645                THEN BEGIN
0048 2-      4646                        CASE TOKENNR OF
0056 --      4647                           SEMICSYM:
0056 3-      4648                              BEGIN
0056 --      4649                                 NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0064 --      4650                                 ADVANCE;
0068 --      4651                                 STATEMENT;
006C --      4652                                 REPSTMLIST
006C -3      4653                              END;
0074 --      4654                           UNTILSYM:
0074 3-      4655                              BEGIN
0074 --      4656                                 NOTSKIP:=TRUE (* ESCAPE ERROR AT SYNCHRSYM *);
0082 --      4657                                 ADVANCE; NEST:=NEST-1
0086 -3      4658                              END
009E -2      4659                        END (* OF CASE *)
0108 -1      4660                     END
0108 1-      4661                ELSE BEGIN
010C --      4662                        ERRMSG26(TOKENNR,'REPEATSTATEMNTLST');
0136 --      4663                        GOTO 9999
0144 -1      4664                     END
0144 -0 A    4665          END; (* OF REPSTMLIST *)
0184 --      4666
0184 -- A    4667          PROCEDURE FORSTMTREM(CONTRTYP: @TYPES);
0044 --      4668          (* <FOR_STATEMENT_REMAINDER> *)
0044 --      4669          VAR EXPTYP: @TYPES;
0000 0- A    4670          BEGIN
000A --      4677             (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4678             IF NOTSKIP THEN
001A --      4679             IF TOKENNR IN (.TOSYM,DOWNTOSYM.)
001A 1-      4680                THEN BEGIN
0054 2-      4681                        CASE TOKENNR OF
0062 --      4682                           TOSYM:
0062 3-      4683                              BEGIN
0062 --      4684                                 ADVANCE;
0066 --      4685                                 EXPRESSION(DUMMYCOD,EXPTYP); COMPARE(EXPTYP, CONTRTYP);
00BC --      4686                                 TERMINAL(DOSYM,DUMMYFD);
00D2 --      4687                                 STATEMENT
00D2 -3      4688                              END;
```

```
00DA --      4689                        DOWNTOSYM:
00DA 3-      4690                           BEGIN
00DA --      4691                              ADVANCE;
00DE --      4692                              EXPRESSION(DUMMYCOD,EXPTYP); COMPARE(EXPTYP, CONTRTYP);
0134 --      4693                              TERMINAL(DOSYM,DUMMYFD);
014A --      4694                              STATEMENT
014A -3      4695                           END
014E -2      4696                     END (* OF CASE *)
01C8 -1      4697                  END
01C8 1-      4698               ELSE BEGIN
01CC --      4699                     ERRMSG26(TOKENNR,'FORSTATEMENTREM  ');
01F6 --      4700                     SYNCHRONIZE; NOTSKIP:=FALSE
01FA -1      4701                  END
0206 -0 A    4702         END; (* OF FORSTMTREM *)
0248 --      4703
0248 -- A    4704         PROCEDURE WITHVARLST;
0040 --      4705         (* <WITH_STATEMENT_VARIABLE_LIST> *)
0040 --      4706         VAR ID: HASHPTR; PTRACT: @ACTIVS; TYPIN, TYPOUT: @TYPES; SECLST: @ACTIVS;
0000 0- A    4707         BEGIN
000A --      4714            (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4715            IF NOTSKIP THEN
001A --      4716            IF TOKENNR IN (.COMMASYM,DOSYM.)
001A 1-      4717               THEN BEGIN
0054 2-      4718                     CASE TOKENNR OF
0062 --      4719                        COMMASYM:
0062 3-      4720                           BEGIN
0062 --      4721                              ADVANCE;
0066 --      4722                              TERMINAL(IDENTIFIER, ID);
007A --      4723                              IF NOTSKIP
007A 4-      4724                                 THEN BEGIN PTRACT:=SEARCHALL(ID, VARID);
00CE --      4725                                         IF PTRACT@.IDKLASS IN (.VARID,FLDID.)
00E2 --      4726                                            THEN TYPIN:=PTRACT@.TYP
012A 5-      4727                                            ELSE BEGIN ERRMSG103(ID);
016A -5      4728                                                       TYPIN:=UNDFZERO END;
018E -4      4729                                    END;
018E --      4730                              VARSELECTS(TRUE, DUMMYSBND, DUMMYCOD,
01C4 --      4731                                         TYPIN, DUMMYCOD, TYPOUT);
01F8 --      4732                              IF TYPOUT@.TYPKLASS<>RECTYP
020C 4-      4733                                 THEN BEGIN IF TYPOUT@.TYPKLASS<>UNDEF
022C --      4734                                               THEN ERRMSG(140,
0240 --      4735                                    'TYPE OF VARIABLE IS NOT A RECORD        ')
0246 -4      4736                                    END
024A 4-      4737                                 ELSE BEGIN SECLST:=TYPOUT@.SECTNS;
0280 --      4738                                            LEVEL:=LEVEL+1;
0298 5-      4739                                            WHILE SECLST<>NIL DO BEGIN
02A4 --      4740                                               NEW(PTRACT,FLDID);
02BC --      4741                                               PTRACT@.IDNR:=SECLST@.IDNR;
02FE --      4742                                               PTRACT@.XREF:=NIL;
0318 --      4743                                               PTRACT@.TYP:=SECLST@.TYP;
035E --      4744                                               PUSH(PTRACT);
0380 -5      4745                                               SECLST:=SECLST@.NXTACT END
03B2 -4      4746                                    END;
03B6 --      4747                              WITHVARLST
03B6 -3      4748                           END;
03BE --      4749                        DOSYM:
03BE --      4750                           ADVANCE
03BE -2      4751                     END (* OF CASE *)
0494 -1      4752                  END
```

```
0494 1-      4753              ELSE BEGIN
0498 --      4754                      ERRMSG26(TOKENNR,'WITHVARIABLELIST ');
04C2 --      4755                      SYNCHRONIZE; NOTSKIP:=FALSE
04C6 -1      4756                   END
04D2 -0 A    4757       END; (* OF WITHVARLST *)
055C --      4758
055C -- A    4759       PROCEDURE UNLABLSTMT;
0040 --      4760       (* <UNLABELED_STATEMENT> *)
0040 --      4761       VAR (* SIMPLE STATEMENT *)
0040 --      4762           ID: HASHPTR;
0044 --      4763           (* COMPOUND STATEMENT *)
0044 --      4764           FIRSTSTMNR: INTEGER; STMCOD, HELPER: CHARVAR;
0050 --      4765           (* IF STATEMENT *)
0050 --      4766           CONDCOD, STMTAIL: CHARVAR; CONDTYP: @TYPES; STM1NR, STM2NR: INTEGER;
0064 --      4767           STACKTRAV: @ACTIVS;
0068 --      4768           (* CASE STATEMENT *)
0068 --      4769           SELTYP: @TYPES; FIRCONSLST, CASCONSLST: @VALUS; MATCHTYP: @TYPES;
0078 --      4770           (* WHILE STATEMENT *)
0078 --      4771           APPENDIX: CHARVAR; WSTMNR: INTEGER;
0080 --      4772           (* FOR STATEMENT *)
0080 --      4773           CONTRTYP ,FROMTYP: @TYPES; PTRACT: @ACTIVS;
008C --      4774           (* WITH STATEMENT *)
008C --      4775           OLDLEVEL: INTEGER; TYPIN, TYPOUT: @TYPES; SECLST: @ACTIVS; NEWX: @XREFS;
00A0 --      4776           (* GOTO STATEMENT *)
00A0 --      4777           LABVAL: INTEGER; LABNR: HASHPTR;
0000 0- A    4778       BEGIN
000A --      4785          (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      4786          IF NOTSKIP THEN
001A --      4787          IF TOKENNR IN (.IDENTIFIER,BEGINSYM,IFSYM,CASESYM,WHILESYM,
001A --      4788                         REPEATSYM,FORSYM,WITHSYM,GOTOSYM,SEMICSYM,ENDSYM,
001A --      4789                         ELSESYM,UNTILSYM.)
001A 1-      4790             THEN BEGIN
0054 2-      4791                     CASE TOKENNR OF
0062 --      4792                        IDENTIFIER:
0062 3-      4793                           BEGIN ID:=TOKENFD;
0082 --      4794                              ADVANCE;
0086 --      4795                              SIMPSTMREM(ID)
00A0 -3      4796                           END;
00A8 --      4797                        BEGINSYM:
00A8 3-      4798                           BEGIN NEST:=NEST+1;
00C0 --      4799                              ADVANCE;
00C4 --      4800                              FIRSTSTMNR:=STMNR+1; (* SPARE PASSING PARAMETERS *)
00D6 --      4801                              STATEMENT;
00DA --      4802                              LOC(STMCOD, 100);
00EE --      4803                              LIT(STMCOD, 5, ' $PHI6789012345');
0108 --      4804                              COMPSTMREM(FIRSTSTMNR, STMCOD);
011C --      4805                              LOC(HELPER, 7); LIT(HELPER, 7, '@ $PHI:89012345');
014A --      4806                              CAT(STMCOD, HELPER, STMCOD); FREE(HELPER);
01B4 --      4807                              WRAP('(',STMCOD,')');
01D0 --      4808                              STMOUT(FIRSTSTMNR-1, STMCOD); FREE(STMCOD)
021C -3      4809                           END;
0224 --      4810                        IFSYM:
0224 3-      4811                           BEGIN
0224 --      4812                              ADVANCE;
0228 --      4813                              LOC(CONDCOD, 50); LOC(HELPER, 7);
0250 --      4814                              LOC(STMCOD, 200); LOC(STMTAIL, 150);
0278 --      4815                              EXPRESSION(CONDCOD, CONDTYP);
028C --      4816                              IF (CONDTYP@.TYPKLASS<>UNDEF) AND
```

```
02B2 --      4817                        (CONDTYP<>TYPBOOLEAN) THEN
02D2 --      4818                        ERRMSG(135,
02DA --      4819                               'TYPE OF OPERAND MUST BE BOOLEAN         ');
02E4 --      4820                     TERMINAL(THENSYM,DUMMYFD);
02FA --      4821                     STM1NR:=STMNR+1;
030C --      4822                     STATEMENT;
0310 --      4823                     ELSECLAUSE(STM2NR);
031C --      4824                     (* PUZZLE IT TOGETHER *)
031C --      4825                     ENVIRON(STMCOD);
0328 --      4826                     LIT(STMTAIL, 4 ,'$IF 56789012345');
0342 --      4827                     APP(STMTAIL, CONDCOD); FREE(CONDCOD);
038E --      4828                     WRAP('(',STMTAIL,')');
03AA --      4829                     CODINTEG(STM1NR, HELPER); CAT(HELPER, STM, HELPER);
040C --      4830                     APP(STMTAIL, HELPER);
0436 --      4831                     WRAP('(',STMTAIL,')');
0452 --      4832                     IF STM2NR<0
0452 --      4833                        THEN LIT(HELPER, 3, '$ID456789012345')
0474 4-      4834                        ELSE BEGIN
047C --      4835                             CODINTEG(STM2NR, HELPER);
0490 --      4836                             CAT(HELPER, STM, HELPER)
04DA -4      4837                             END;
04DE --      4838                     APP(STMTAIL, HELPER); FREE(HELPER);
052A --      4839                     WRAP('(',STMTAIL,')');
0546 --      4840                     APPENV(STMTAIL);
0552 --      4841                     APP(STMCOD, STMTAIL); FREE(STMTAIL);
059E --      4842                     WRAP('(',STMCOD,')');
05BA --      4843                     STMOUT(STM1NR-1, STMCOD); FREE(STMCOD)
0606 -3      4844                  END;
060E --      4845               CASESYM:
060E 3-      4846                  BEGIN NEST:=NEST+1;
0626 --      4847                     ADVANCE;
062A --      4848                     EXPRESSION(DUMMYCOD,SELTYP);
0640 --      4849                     IF SELTYP@.TYPKLASS<>SCALTYP
0654 4-      4850                        THEN BEGIN IF SELTYP@.TYPKLASS<>UNDEF
0674 --      4851                                      THEN ERRMSG(144,
0688 --      4852                                'SELECTOR TYPE MUST BE SCALAR OR SUBRANGE');
0692 -4      4853                             MATCHTYP:=UNDFZERO END
06B6 --      4854                        ELSE MATCHTYP:=SELTYP;
06D8 --      4855                     TERMINAL(OFSYM,DUMMYFD);
06EE --      4856                     CASEELEMNT(FIRCONSLST, MATCHTYP);
0718 --      4857                     CASELEMLST(FIRCONSLST, CASCONSLST, MATCHTYP)
075C -3      4858                  END;
0764 --      4859               WHILESYM:
0764 3-      4860                  BEGIN
0764 --      4861                     ADVANCE;
0768 --      4862                     LOC(HELPER, 7); LOC(APPENDIX, 20);
0790 --      4863                     LOC(CONDCOD, 50);
07A4 --      4864                     LOC(STMTAIL, 150); LOC(STMCOD, 200);
07CC --      4865                     EXPRESSION(CONDCOD,CONDTYP);
07E0 --      4866                     IF (CONDTYP@.TYPKLASS<>UNDEF) AND
0806 --      4867                        (CONDTYP<>TYPBOOLEAN) THEN
0826 --      4868                        ERRMSG(135,
082E --      4869                               'TYPE OF OPERAND MUST BE BOOLEAN         ');
0838 --      4870                     TERMINAL(DOSYM,DUMMYFD);
084E --      4871                     WSTMNR:=STMNR;
085C --      4872                     STATEMENT;
0860 --      4873                     (* PUZZLE IT TOGETHER *)
0860 --      4874                     ENVIRON(STMCOD);
```

```
086C --      4875                        LIT(STMTAIL, 4, '$IF 56789012345');
0886 --      4876                        APP(STMTAIL, CONDCOD); FREE(CONDCOD);
08D2 --      4877                        WRAP('(',STMTAIL,')');
08EE --      4878                        CODINTEG(WSTMNR, APPENDIX); CAT(APPENDIX, STM, APPENDIX);
0950 --      4879                        LIT(HELPER, 5, ' $PHI6789012345');
096A --      4880                        APP(APPENDIX, HELPER); WRAP('(',APPENDIX,')');
09B0 --      4881                        CODINTEG(WSTMNR+1, HELPER); CAT(HELPER, STM, HELPER);
0A16 --      4882                        CAT(APPENDIX, HELPER, APPENDIX); FREE(HELPER);
0A80 --      4883                        WRAP('(',APPENDIX,')');
0A9C --      4884                        APP(STMTAIL, APPENDIX); FREE(APPENDIX);
0AE8 --      4885                        WRAP('(',STMTAIL,')');
0B04 --      4886                        APPENV(STMTAIL);
0B10 --      4887                        APP(STMCOD, STMTAIL); FREE(STMTAIL);
0B5C --      4888                        WRAP('(',STMCOD,')');
0B78 --      4889                        STMOUT(WSTMNR, STMCOD); FREE(STMCOD)
0BC0 -3      4890                     END;
0BC8 --      4891                  REPEATSYM:
0BC8 3-      4892                     BEGIN NEST:=NEST+1;
0BE0 --      4893                        ADVANCE;
0BE4 --      4894                        STATEMENT;
0BE8 --      4895                        REPSTMLIST;
0BEC --      4896                        EXPRESSION(DUMMYCOD,CONDTYP);
0C02 --      4897                        IF (CONDTYP@.TYPKLASS<>UNDEF) AND
0C28 --      4898                           (CONDTYP<>TYPBOOLEAN) THEN
0C48 --      4899                            ERRMSG(135,
0C50 --      4900                                   'TYPE OF OPERAND MUST BE BOOLEAN         ');
0C5A -3      4901                     END;
0C5E --      4902                  FORSYM:
0C5E 3-      4903                     BEGIN
0C5E --      4904                        ADVANCE;
0C62 --      4905                        TERMINAL(IDENTIFIER,ID);
0C76 4-      4906                        IF NOTSKIP THEN BEGIN
0C86 --      4907                           PTRACT:=SEARCHALL(ID, VARID);
0CCA --      4908                           IF PTRACT@.IDKLASS<>VARID
0CDE 5-      4909                              THEN BEGIN ERRMSG103(ID);
0D08 -5      4910                                      CONTRTYP:=UNDFZERO END
0D2C --      4911                              ELSE IF PTRACT@.TYP@.TYPKLASS<>SCALTYP
0D58 5-      4912                                      THEN BEGIN ERRMSG(143,
0D6C --      4913                                'ILLEGAL TYPE OF LOOP CONTROL VARIABLE   ');
0D76 -5      4914                                           CONTRTYP:=UNDFZERO END
0D9A --      4915                                      ELSE CONTRTYP:=PTRACT@.TYP
0DB2 -4      4916                           END ELSE CONTRTYP:=UNDFZERO;
0DF8 --      4917                        TERMINAL(BECOMES,DUMMYFD);
0E0E --      4918                        EXPRESSION(DUMMYCOD,FROMTYP); COMPARE(CONTRTYP, FROMTYP);
0E64 --      4919                        FORSTMTREM(CONTRTYP)
0E82 -3      4920                     END;
0E8A --      4921                  WITHSYM:
0E8A 3-      4922                     BEGIN
0E8A --      4923                        ADVANCE; OLDLEVEL:=LEVEL;
0E9C --      4924                        TERMINAL(IDENTIFIER, ID);
0EB0 --      4925                        IF NOTSKIP
0EB0 4-      4926                           THEN BEGIN PTRACT:=SEARCHALL(ID, VARID);
0F04 --      4927                                   IF PTRACT@.IDKLASS IN (.VARID,FLDID.)
0F18 --      4928                                      THEN TYPIN:=PTRACT@.TYP
0F60 5-      4929                                      ELSE BEGIN ERRMSG103(ID);
0FA0 -5      4930                                                 TYPIN:=UNDFZERO END;
0FC4 -4      4931                                END;
0FC4 --      4932                        VARSELECTS(TRUE, DUMMYSBND, DUMMYCOD,
```

```
0FFA --    4933                                    TYPIN, DUMMYCOD, TYPOUT);
102E --    4934                              IF TYPOUT@.TYPKLASS<>RECTYP
1042 4-    4935                                 THEN BEGIN IF TYPOUT@.TYPKLASS<>UNDEF
1062 --    4936                                               THEN ERRMSG(140,
1076 --    4937                                         'TYPE OF VARIABLE IS NOT A RECORD        ')
107C -4    4938                                         END
1080 4-    4939                                    ELSE BEGIN SECLST:=TYPOUT@.SECTNS;
10B6 --    4940                                            LEVEL:=LEVEL+1;
10CE 5-    4941                                            WHILE SECLST<>NIL DO BEGIN
10DA --    4942                                               NEW(PTRACT,FLDID);
10F2 --    4943                                               PTRACT@.IDNR:=SECLST@.IDNR;
1134 --    4944                                               NEW(NEWX); NEWX@.OCC:=SCNR;
1160 --    4945                                               NEWX@.NXTREF:=NIL;PTRACT@.XREF:=NEWX;
11AC --    4946                                               PTRACT@.TYP:=SECLST@.TYP;
11F2 --    4947                                               PUSH(PTRACT);
1214 -5    4948                                               SECLST:=SECLST@.NXTACT END
1246 -4    4949                                         END;
124A --    4950                                 WITHVARLST;
124E --    4951                                 STATEMENT;
1252 --    4952                                 LEVEL:=OLDLEVEL+1; POP; LEVEL:=OLDLEVEL
1268 -3    4953                              END;
127A --    4954                           GOTOSYM:
127A 3-    4955                              BEGIN
127A --    4956                                 ADVANCE;
127E --    4957                                 TERMINAL(UNSGINTEG,LABVAL);
1292 --    4958                                 LABNR:=CONVPSID(LABVAL,'LABEL');
12BE --    4959                                 PTRACT:=SEARCHALL(LABNR, SCALID)
12DE -3    4960                              END;
1304 --    4961                           SEMICSYM,ENDSYM,ELSESYM,UNTILSYM:
1304 --    4962                              (* EPSILON PRODUCTION *)
1304 3-    4963                              BEGIN
1304 --    4964                                 LOC(STMCOD, 15);
1318 --    4965                                 LIT(STMCOD, 3, '$ID456789012345');
1332 --    4966                                 STMOUT(STMNR, STMCOD); FREE(STMCOD)
1380 -3    4967                              END
1384 -2    4968                        END (* OF CASE *)
1468 -1    4969                     END
1468 1-    4970                  ELSE BEGIN
146C --    4971                           ERRMSG26(TOKENNR,'UNLABELEDSTATEMNT');
1496 --    4972                           SYNCHRONIZE; NOTSKIP:=FALSE
149A -1    4973                        END;
14A6 -0 A  4974            END; (* OF UNLABLSTMT *)
1600 --    4975
1600 -- A  4976            PROCEDURE STATEMENT;
0040 --    4977            (* <STATEMENT> *)
0040 --    4978            VAR LABVAL: INTEGER; LABNR: HASHPTR; PTRACT: @ACTIVS;
0000 0- A  4979            BEGIN
000A --    4986               (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    4987               IF NOTSKIP THEN
001A --    4988               IF TOKENNR IN (.UNSGINTEG,IDENTIFIER,SEMICSYM,ENDSYM,CASESYM,
001A --    4989                              BEGINSYM,IFSYM,WHILESYM,REPEATSYM,FORSYM,WITHSYM,
001A --    4990                              GOTOSYM,ELSESYM,UNTILSYM.)
001A 1-    4991                  THEN BEGIN STMNR:=STMNR+1;
006C 2-    4992                          CASE TOKENNR OF
007A --    4993                             UNSGINTEG:
007A 3-    4994                                BEGIN LABVAL:=TOKENFD;
0088 --    4995                                   LABNR:=CONVPSID(LABVAL, 'LABEL');
00B4 --    4996                                   PTRACT:=SEARCHALL(LABNR, SCALID);
```

```
00F6 --      4997                              (* FLAG THAT IT IS AN ACTUAL LOCATION *)
00F6 --      4998                              PTRACT@.VALUU:=1;
0112 --      4999                              ADVANCE;
0116 --      5000                              TERMINAL(COLONSYM,DUMMYFD);
012C --      5001                              UNLABLSTMT
012C -3      5002                           END;
0134 --      5003                        IDENTIFIER,SEMICSYM,ENDSYM,CASESYM,BEGINSYM,IFSYM,
0134 --      5004                        WHILESYM,REPEATSYM,FORSYM,WITHSYM,GOTOSYM,ELSESYM,
0134 --      5005                        UNTILSYM:
0134 --      5006                           UNLABLSTMT
0134 -2      5007                     END (* OF CASE *)
023A -1      5008                  END
023A 1-      5009               ELSE BEGIN
023E --      5010                     ERRMSG26(TOKENNR,'STATEMENT        ');
0268 --      5011                     SYNCHRONIZE; NOTSKIP:=FALSE
026C -1      5012                  END
0278 -0 A    5013      END; (* OF STATEMENT *)
02C8 --      5014
02C8 -- A    5015      PROCEDURE BLOCK;
0040 --      5016      (* <BLOCK> *)
0040 --      5017      VAR PTRACT: @ACTIVS; STACKTRAV: @ACTIVS; COMPON: @TYPES;
004C --      5018          NOTTHRU: BOOLEAN; (* FLAGS IF CURRENT LEVEL EXHAUSTED *)
0050 --      5019          BODYNR, BLKLEVNR, DIMNR: INTEGER;
005C --      5020          COMPSTMCOD, HELPER, OMEGAS, LEFTPARAS, LEFTP, RIGHTP, TUPINIT: CHARVAR;
0000 0- A    5021      BEGIN
000A --      5028         (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --      5029         IF NOTSKIP THEN
001A --      5030         IF TOKENNR IN (.LABELSYM,CONSTSYM,TYPESYM,VARSYM,PROCSYM,FUNCSYM,
001A --      5031                        BEGINSYM.)
001A 1-      5032            THEN BEGIN
0054 --      5033                    LABELDCL;
0058 --      5034                    CONSDEFPRT;
005C --      5035                    ALLOWFW:=TRUE; TYPEDEFPRT;
006E --      5036                    (* SEE IF ALL FORWARD REFERENCES HAVE BEEN SATISFIED *)
006E --      5037                    WHILE FORWTYPS<>NIL DO
0080 2-      5038                       BEGIN
0080 --      5039                          PTRACT:=SEARCHALL(FORWTYPS@.LOCTYP@.PTRIDNR, TYPID);
00F2 --      5040                          PTRACT@.XREF@.OCC:=FORWTYPS@.LOCSCNR;
013C --      5041                          IF PTRACT@.IDKLASS=TYPID
0150 --      5042                             THEN FORWTYPS@.LOCTYP@.REFERTYP:=PTRACT@.TYP
019E 3-      5043                             ELSE BEGIN
01C0 --      5044                                  FORWTYPS@.LOCTYP@.REFERTYP:=UNDFZERO;
0212 --      5045                                  ERRMSG103(FORWTYPS@.LOCTYP@.PTRIDNR)
025A -3      5046                                  END;
025E --      5047                          FORWTYPS:=FORWTYPS@.NEXTF
0278 -2      5048                       END; ALLOWFW:=FALSE;
02AC --      5049                    VARDCLPART;
02B0 --      5050                    PROCFUNPRT;
02B4 --      5051                    IF NOT NOTSKIP THEN GOTO 9999; (* DONT ASSUME BEGIN THEN *)
02D6 --      5052                    TERMINAL(BEGINSYM,DUMMYFD);
02EC --      5053                    STMNR:=STMNR+1; NEST:=NEST+1;
031C --      5054                    (* PREPARE EXIT-EXPRESSION  AND INTITIALIZATIONS  *)
031C --      5055                    (* COMPSTMCOD WILL HOLD EXIT-EXPRESSION          *)
031C --      5056                    (* LEFTPARAS HOLDS ALL LEFT PARENTHESIS NEEDED TO *)
031C --      5057                    (*    PREFIX IN FRONT OF THE COMPLETED COMPSTMCOD *)
031C --      5058                    (* OMEGAS HOLDS THE REST OF THE INITIALIZATION    *)
031C --      5059                    LOC(HELPER, 50); LOC(OMEGAS, 50); LOC(TUPINIT, 50);
0358 --      5060                    LOC(LEFTP, 1); LOC(RIGHTP, 1); LOC(LEFTPARAS, 50);
```

```
0394 --    5061                    LIT(LEFTP, 1, '(23456789012345');
03AE --    5062                    LIT(RIGHTP, 1, ')23456789012345');
03C8 --    5063                    LOC(COMPSTMCOD, 100);
03DC --    5064                    IF LEVEL>1
03DC --    5065                       THEN BLKLEVNR:=LEVEL
03EE --    5066                       ELSE BLKLEVNR:=0; (* INCLUDE PREDEFINED VARIABLES *)
0406 --    5067                    COMPSTMCOD@.CURLG:=0;
0420 --    5068                    OMEGAS@.CURLG:=0; LEFTPARAS@.CURLG:=0;
0454 --    5069                    STACKTRAV:=STACK;
0478 --    5070                    (* SINCE THERE IS ALWAYS ONE ACTIVE SYMBOL WE CAN ... *)
0478 --    5071                    NOTTHRU:=(STACKTRAV@.LEVNR>=BLKLEVNR);
04B4 --    5072                    WHILE NOTTHRU DO
04BE 2-    5073                       BEGIN
04BE 3-    5074                          CASE STACKTRAV@.IDKLASS OF
04DA --    5075                          SCALID, TYPID, PROCID, FUNCID:;
04DE --    5076                          VARID, FLDID:
04DE 4-    5077                             BEGIN
04DE --    5078                                CODIDENT(STACKTRAV, HELPER);
0508 --    5079                                IF COMPSTMCOD@.CURLG<>0
051C --    5080                                   THEN APP(COMPSTMCOD, LITCOMMA)
0554 --    5081                                   ELSE LIT(COMPSTMCOD, 2, '@ 3456789012345');
0576 --    5082                                APP(COMPSTMCOD, HELPER);
05A0 --    5083                                (* CHECK IF VARIABLE A PARAMETER *)
05A0 --    5084                                IF STACKTRAV@.PARM=VARB
05B4 --    5085                                THEN (* INITIALIZE THEM AS OMEGA *)
05C0 5-    5086                                CASE STACKTRAV@.TYP@.TYPKLASS OF
05F0 --    5087                                SCALTYP, UNDEF:
05F0 6-    5088                                   BEGIN
05F0 --    5089                                      APP(OMEGAS, OMEGA);
0620 --    5090                                      APP(LEFTPARAS, LEFTP); APP(OMEGAS, RIGHTP);
0674 -6    5091                                   END;
0678 --    5092                                ARRTYP:
0678 6-    5093                                   BEGIN
0678 --    5094                                      (* DETERMINE THE NUMBER OF DIMENSIONS *)
0678 --    5095                                      DIMNR:=0;
067E --    5096                                      COMPON:=STACKTRAV@.TYP;
06B0 --    5097                                      WHILE COMPON@.TYPKLASS=ARRTYP DO
06D0 7-    5098                                         BEGIN DIMNR:=DIMNR+1;
06DC --    5099                                            COMPON:=COMPON@.COMPTYP
06F0 -7    5100                                         END;
0712 --    5101                                      (* BUILD TUPINIT *)
0712 --    5102                                      LIT(TUPINIT, 9, '$TUPINIT 012345');
072C --    5103                                      CODINTEG(DIMNR, HELPER);
0740 --    5104                                      APP(TUPINIT, HELPER);
076A --    5105                                      WRAP('(',TUPINIT,')');
0786 --    5106                                      COMPON:=STACKTRAV@.TYP;
07B8 --    5107                                      WHILE COMPON@.TYPKLASS=ARRTYP DO
07D8 7-    5108                                         BEGIN
07D8 --    5109                                            IF COMPON@.INDXTYP@.TYPKLASS=SCALTYP THEN
080C --    5110                                            CODINTEG(COMPON@.INDXTYP@.HIGBND-
0834 --    5111                                                    COMPON@.INDXTYP@.LOWBND+1, HELPER);
0878 --    5112                                            APP(TUPINIT, HELPER);
08A2 --    5113                                            WRAP('(',TUPINIT,')');
08BE --    5114                                            COMPON:=COMPON@.COMPTYP
08D2 -7    5115                                         END;
08F4 --    5116                                      APP(OMEGAS, TUPINIT);
091E --    5117                                      APP(LEFTPARAS, LEFTP); APP(OMEGAS, RIGHTP)
096E -6    5118                                END;
```

```
0976 --    5119                                RECTYP:;
097A --    5120                                PTRTYP:
097A --    5121                                   IF (STACKTRAV@.IDNR=SPINPUT) OR
09A6 --    5122                                      (STACKTRAV@.IDNR=SPOUTPUT) THEN
09DA 6-    5123                                      BEGIN
09DA --    5124                                         APP(OMEGAS, OMEGA);
0A0A --    5125                                         APP(LEFTPARAS, LEFTP); APP(OMEGAS, RIGHTP)
0A5A -6    5126                                      END
0A5E -5    5127                                END (*TYPE-CASE*)
0A90 --    5128                                ELSE (* TREAT THEM AS VALUE PARAMETERS *)
0A94 5-    5129                                BEGIN
0A94 --    5130                                   CODIDENT(STACKTRAV, HELPER);
0ABE --    5131                                   CAT(HELPER, VALPREF, HELPER);
0B0C --    5132                                   APP(OMEGAS, HELPER);
0B36 --    5133                                   APP(LEFTPARAS, LEFTP); APP(OMEGAS, RIGHTP)
0B86 -5    5134                                END
0B8A -4    5135                             END
0B8A -3    5136                          END; (*ID-CASE*)
0BC0 --    5137                          STACKTRAV:=STACKTRAV@.NXTACT;
0BF2 --    5138                          IF STACKTRAV<>NIL
0BF2 --    5139                             THEN NOTTHRU:=(STACKTRAV@.LEVNR>=BLKLEVNR)
0C24 --    5140                             ELSE NOTTHRU:=FALSE
0C3E -2    5141                       END;
0C48 --    5142                    FREE(TUPINIT); FREE(LEFTP); FREE(RIGHTP);
0CAE --    5143                    IF COMPSTMCOD@.CURLG<>0
0CC2 2-    5144                       THEN BEGIN
0CCE --    5145                               LIT(HELPER, 5, ':$PHI6789012345');
0CE8 --    5146                               APP(COMPSTMCOD, HELPER);
0D12 --    5147                               WRAP('(',COMPSTMCOD,')')
0D2A -2    5148                            END
0D2E --    5149                       ELSE LIT(COMPSTMCOD, 5, ' $PHI6789012345');
0D4C --    5150                    BODYNR:=STMNR;
0D5A --    5151                    IF LEVEL=1 THEN PROGSTMNR:=BODYNR;
0D7A --    5152                    STATEMENT;
0D7E --    5153                    COMPSTMREM(BODYNR+1, COMPSTMCOD);
0D96 --    5154                    CAT(COMPSTMCOD, LEFTPARAS, COMPSTMCOD); FREE(LEFTPARAS);
0E00 --    5155                    APP(COMPSTMCOD, OMEGAS); FREE(OMEGAS);
0E4C --    5156                    LIT(HELPER, 7, '@ $PHI:89012345');
0E66 --    5157                    CAT(COMPSTMCOD, HELPER, COMPSTMCOD); FREE(HELPER);
0ED0 --    5158                    WRAP('(', COMPSTMCOD, ')');
0EEC --    5159                    STMOUT(BODYNR, COMPSTMCOD);
0F16 --    5160                    FREE(COMPSTMCOD)
0F34 -1    5161                 END
0F38 1-    5162              ELSE BEGIN
0F3C --    5163                      ERRMSG26(TOKENNR,'BLOCK            ');
0F66 --    5164                      SYNCHRONIZE; NOTSKIP:=FALSE
0F6A -1    5165                 END
0F76 -0 A  5166        END; (* OF BLOCK *)
1034 --    5167
1034 -- A  5168        PROCEDURE PROGRAMM;
0040 --    5169        (* <PROGRAM> *)
0040 --    5170        VAR FIRSTFILE: HASHPTR; FILELIST: @IDLST; PTRACT: @ACTIVS;
004C --    5171            PROGCOD, HELPER: CHARVAR;
0000 0- A  5172        BEGIN
000A --    5179           (*$L+ INVISIBLE DEBUG COMMANDS HERE *)
000A --    5180           IF TOKENNR=PROGSYM
000A 1-    5181              THEN BEGIN
001C --    5182                      ADVANCE;
```

```
0020 --      5183                    TERMINAL(IDENTIFIER,PROGID);
0034 --      5184                    WRITELN(SPUNCH, '* PASCAL K CODE MODULE FOR ',
004A --      5185                            HASHNAME(.PROGID.));
007C --      5186                    WRITELN(SPUNCH,'*');
0096 --      5187                    TERMINAL(LPARASYM,DUMMYFD);
00AC --      5188                    TERMINAL(IDENTIFIER,FIRSTFILE);
00C0 --      5189                    IDENTLIST(FIRSTFILE,FILELIST);
00E6 --      5190                    IF NOTSKIP THEN
00F6 2-      5191                       WHILE FILELIST<>NIL DO BEGIN
0102 3-      5192                          IF FILELIST@.ID=SPINPUT THEN BEGIN
0128 --      5193                             NEW(PTRACT,VARID); NEW(PTRACT@.TYP,PTRTYP);
016C 4-      5194                             WITH PTRACT@,TYP@ DO BEGIN
0198 --      5195                                IDNR:=SPINPUT; LEVNR:=0; PARM:=VARB; NEW(XREF);
01D4 --      5196                                XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL;
020A --      5197                                REFERTYP:=TYPINTEGER;
022E -4      5198                                PTRIDNR:=SPINTEGER END;
024E -3      5199                                IF SEARCHONLY(SPINPUT) THEN PUSH(PTRACT) END
029E 3-      5200                          ELSE IF FILELIST@.ID=SPOUTPUT THEN BEGIN
02C8 --      5201                             NEW(PTRACT,VARID); NEW(PTRACT@.TYP,PTRTYP);
030C 4-      5202                             WITH PTRACT@,TYP@ DO BEGIN
0338 --      5203                                IDNR:=SPOUTPUT; LEVNR:=0; PARM:=VARB; NEW(XREF);
0374 --      5204                                XREF@.OCC:=SCNR; XREF@.NXTREF:=NIL;
03AA --      5205                                REFERTYP:=TYPINTEGER;
03CE -4      5206                                PTRIDNR:=SPINTEGER END;
03EE -3      5207                                IF SEARCHONLY(SPOUTPUT) THEN PUSH(PTRACT) END
043E --      5208                          ELSE ERRMSG(398,
044A --      5209                             'ONLY INPUT/OUTPUT SUPPORTED             ');
0454 -2      5210                          FILELIST:=FILELIST@.NXTID END;
048A --      5211                    TERMINAL(RPARASYM,DUMMYFD);
04A0 --      5212                    LEVEL:=1;
04AE --      5213                    TERMINAL(SEMICSYM,DUMMYFD);
04C4 --      5214                    BLOCK; LEVEL:=0; POP;
04D8 --      5215                    (* IT'S ALL DONE *)
04D8 --      5216                    LOC(PROGCOD, 40); LOC(HELPER, 10);
0500 --      5217                    CODINTEG(PROGSTMNR, HELPER);
051A --      5218                    LIT(PROGCOD, 15, '$PROGRAM=(($STM');
0534 --      5219                    APP(PROGCOD, HELPER);
055E --      5220                    LIT(HELPER, 10, ' $ID)$ID).12345');
0578 --      5221                    APP(PROGCOD, HELPER); FREE(HELPER);
05C4 --      5222                    OUT(PROGCOD);
05E6 --      5223                    IF TOKENNR<>PERIODSYM THEN
05F8 --      5224                       ERRMSG(27,'PROGRAM MUST END WITH A PERIOD          ')
0606 -1      5225                 END
060A 1-      5226              ELSE BEGIN
060E --      5227                    ERRMSG26(TOKENNR,'PROGRAM          ');
0638 --      5228                    GOTO 9999
0646 -1      5229                 END
0646 -0 A    5230        END; (* OF PROGRAMM *)
072C --      5231
072C -- A    5232        PROCEDURE INIT;
0040 --      5233           VAR I: INTEGER; C: CHAR; PTRACT: @ACTIVS;
0000 0- A    5234           BEGIN
000A --      5235              SCNR:=0; LNPOS:=0; SWEOLN:=FALSE;
001C --      5236              FOR I:=1 TO MAXLNL DO LINE(.I.):=' ';
0060 --      5237              CH:=CHR(ORDEOLNCH);
006E --      5238
006E --      5239           KEYW(. 1.):='AND       '; KEYW(. 2.):='ARRAY     '; KEYW(. 3.):='BEGIN     ';
0080 --      5240           KEYW(. 4.):='CASE      '; KEYW(. 5.):='CONST     '; KEYW(. 6.):='DIV       ';
```

```
0092 --      5241          KEYW(. 7.):='DO        '; KEYW(. 8.):='DOWNTO    '; KEYW(. 9.):='ELSE      ';
00A4 --      5242          KEYW(.10.):='END       '; KEYW(.11.):='FILE      '; KEYW(.12.):='FOR       ';
00B6 --      5243          KEYW(.13.):='FUNCTION  '; KEYW(.14.):='GOTO      '; KEYW(.15.):='IF        ';
00C8 --      5244          KEYW(.16.):='IN        '; KEYW(.17.):='LABEL     '; KEYW(.18.):='MOD       ';
00DA --      5245          KEYW(.19.):='NIL       '; KEYW(.20.):='NOT       '; KEYW(.21.):='OF        ';
00EC --      5246          KEYW(.22.):='OR        '; KEYW(.23.):='PACKED    '; KEYW(.24.):='PROCEDURE ';
00FE --      5247          KEYW(.25.):='PROGRAM   '; KEYW(.26.):='RECORD    '; KEYW(.27.):='REPEAT    ';
0110 --      5248          KEYW(.28.):='SET       '; KEYW(.29.):='THEN      '; KEYW(.30.):='TO        ';
0122 --      5249          KEYW(.31.):='TYPE      '; KEYW(.32.):='UNTIL     '; KEYW(.33.):='VAR       ';
0134 --      5250          KEYW(.34.):='WHILE     '; KEYW(.35.):='WITH      ';
0140 --      5251          TOKW(.40.):='IDENTIFIER';
0146 --      5252          TOKW(.46.):='(         '; TOKW(.47.):=')         '; TOKW(.48.):='(. / [    ';
0158 --      5253          TOKW(.49.):='.) / ]    '; TOKW(.50.):=';         '; TOKW(.51.):=',         ';
016A --      5254          TOKW(.52.):='@         '; TOKW(.53.):='=         '; TOKW(.54.):='.         ';
017C --      5255          TOKW(.55.):='..        '; TOKW(.56.):=':         '; TOKW(.57.):=':=        ';
018E --      5256          TOKW(.58.):='STRING    '; TOKW(.59.):='UNSINTEGER'; TOKW(.60.):='UNSIGNREAL';
01A0 --      5257          TOKW(.61.):='PLUS/MINUS'; TOKW(.62.):='MULTIPOPER'; TOKW(.63.):='RELATNOPER';
01B2 --      5258
01B2 --      5259             FOR C:=CHR(LOWORD) TO CHR(HIGORD) DO CHRSEL(.C.):=OTHERCHR;
020A --      5260             FOR C:='A' TO 'Z' DO CHRSEL(.C.):=LETTER;
0264 --      5261             FOR C:='0' TO '9' DO CHRSEL(.C.):=DIGIT;
02BE --      5262             CHRSEL(.'.'.):=PERIODCHR; CHRSEL(.'E'.):=ELETTER  ;
02CE --      5263             CHRSEL(.'+'.):=PLUS     ; CHRSEL(.'-'.):=MINUS    ;
02DE --      5264             CHRSEL(.'*'.):=STAR     ; CHRSEL(.'/'.):=SLASH    ;
02EE --      5265             CHRSEL(.'('.):=LPARACHR ; CHRSEL(.')'.):=RPARACHR ;
02FE --      5266             CHRSEL(.'['.):=LBRACKCHR; CHRSEL(.']'.):=RBRACKCHR;
030E --      5267             CHRSEL(.';'.):=SEMICCHR ; CHRSEL(.','.):=COMMACHR ;
031E --      5268             CHRSEL(.'@'.):=AMPERSAND; CHRSEL(.'='.):=EQUALCHR ;
032E --      5269             CHRSEL(.'<'.):=LESSCHR  ; CHRSEL(.'>'.):=GREATRCHR;
033E --      5270             CHRSEL(.':'.):=COLONCHR ; CHRSEL(.''''.):=QUOTE   ;
034E --      5271             CHRSEL(.' '.):=BLANK    ; CHRSEL(.'{'.):=LBRACECHR;
035E --      5272             CHRSEL(.CHR(ORDEOLNCH).):=EOLNCHR;
0366 --      5273             CHRSEL(.CHR(ORDDBLDOTCH).):=DBLDOTCHR;
036E --      5274             FOR I:=0 TO 9 DO DIGCHR(.I.):=CHR(I+ORD('0'));
03BA --      5275
03BA --      5276             TRANSIT(.1,1,1.):= 1; TRANSIT(.1,2,1.):= 2; TRANSIT(.1,3,1.):= 4;
03D2 --      5277                                   TRANSIT(.1,4,1.):= 0; TRANSIT(.1,5,1.):= 0;
03DE --      5278             TRANSIT(.2,1,1.):= 3; TRANSIT(.2,2,1.):= 0; TRANSIT(.2,3,1.):= 0;
03F2 --      5279                                   TRANSIT(.2,4,1.):= 0; TRANSIT(.2,5,1.):= 0;
03FE --      5280             TRANSIT(.3,1,1.):= 3; TRANSIT(.3,2,1.):= 0; TRANSIT(.3,3,1.):= 4;
0414 --      5281                                   TRANSIT(.3,4,1.):= 0; TRANSIT(.3,5,1.):= 0;
0420 --      5282             TRANSIT(.4,1,1.):= 6; TRANSIT(.4,2,1.):= 0; TRANSIT(.4,3,1.):= 0;
0434 --      5283                                   TRANSIT(.4,4,1.):= 5; TRANSIT(.4,5,1.):= 0;
0442 --      5284             TRANSIT(.5,1,1.):= 6; TRANSIT(.5,2,1.):= 0; TRANSIT(.5,3,1.):= 0;
0456 --      5285                                   TRANSIT(.5,4,1.):= 0; TRANSIT(.5,5,1.):= 0;
0462 --      5286             TRANSIT(.6,1,1.):= 6; TRANSIT(.6,2,1.):= 0; TRANSIT(.6,3,1.):= 0;
0476 --      5287                                   TRANSIT(.6,4,1.):= 0; TRANSIT(.6,5,1.):= 0;
0482 --      5288             TRANSIT(.1,1,2.):= 1; TRANSIT(.1,2,2.):= 4; TRANSIT(.1,3,2.):= 3;
049A --      5289                                   TRANSIT(.1,4,2.):= 2; TRANSIT(.1,5,2.):= 2;
04AA --      5290             TRANSIT(.2,1,2.):= 3; TRANSIT(.2,2,2.):= 5; TRANSIT(.2,3,2.):= 5;
04C2 --      5291                                   TRANSIT(.2,4,2.):= 5; TRANSIT(.2,5,2.):= 5;
04D2 --      5292             TRANSIT(.3,1,2.):= 4; TRANSIT(.3,2,2.):= 4; TRANSIT(.3,3,2.):= 4;
04EA --      5293                                   TRANSIT(.3,4,2.):= 4; TRANSIT(.3,5,2.):= 4;
04FA --      5294             TRANSIT(.4,1,2.):= 4; TRANSIT(.4,2,2.):= 6; TRANSIT(.4,3,2.):= 6;
0512 --      5295                                   TRANSIT(.4,4,2.):= 4; TRANSIT(.4,5,2.):= 6;
0522 --      5296             TRANSIT(.5,1,2.):= 4; TRANSIT(.5,2,2.):= 6; TRANSIT(.5,3,2.):= 6;
053A --      5297                                   TRANSIT(.5,4,2.):= 6; TRANSIT(.5,5,2.):= 6;
054A --      5298             TRANSIT(.6,1,2.):= 4; TRANSIT(.6,2,2.):= 4; TRANSIT(.6,3,2.):= 4;
```

```
0562 --      5299                                    TRANSIT(.6,4,2.):= 4; TRANSIT(.6,5,2.):= 4;
0572 --      5300
0572 --      5301            ERRCNT:=0;
057E --      5302            STRGPTR:=1;
058C --      5303
058C --      5304            FOR I:=0 TO MAXNRID DO
0590 11      5305                BEGIN HASHNAME(.I.):='??????????'; SRCHTAB(.I.):=NIL END;
05FE --      5306            FREESPACE:=NRBUCK;
060C --      5307
060C --      5308            NOTSKIP:=TRUE;
061A --      5309            SYNCHRSYM:=(.SEMICSYM, ENDSYM, ELSESYM, UNTILSYM.);
0626 --      5310            DONTSKIPSYM:=(.PROCSYM, FUNCSYM, BEGINSYM, RECORDSYM, CASESYM,REPEATSYM.);
0638 --      5311            LEVEL:=0; STMNR:=0; NEST:=0; STACK:=NIL; ATTR:=NIL;
0674 --      5312
0674 --      5313            (* PRE-DEFINED IDENTIFIERS AND NAMES *)
0674 --      5314            IDREG:='INTEGER   '; SPINTEGER:=HASH; IDREG:='REAL      '; SPREAL :=HASH;
06C8 --      5315            IDREG:='BOOLEAN   '; SPBOOLEAN:=HASH; IDREG:='CHAR      '; SPCHAR :=HASH;
071C --      5316            IDREG:='FALSE     '; SPFALSE  :=HASH; IDREG:='TRUE      '; SPTRUE :=HASH;
0770 --      5317            IDREG:='INPUT     '; SPINPUT  :=HASH; IDREG:='LABEL     '; SPLABEL:=HASH;
07C4 --      5318            IDREG:='OUTPUT    '; SPOUTPUT :=HASH; IDREG:='FILE      '; SPFILE :=HASH;
0818 --      5319            IDREG:='ORD       '; SPORD    :=HASH; IDREG:='SET       '; SPSET  :=HASH;
086C --      5320            IDREG:='CHR       '; SPCHR    :=HASH; IDREG:='GET       '; SPGET  :=HASH;
08C0 --      5321            IDREG:='PUT       '; SPPUT    :=HASH; IDREG:='NEW       '; SPNEW  :=HASH;
0914 --      5322            IDREG:='FORWARD   '; SPFORWARD:=HASH; IDREG:='$CONSTANT '; SPCONST:=HASH;
0968 --      5323            IDREG:='SUCC      '; SPSUCC   :=HASH; IDREG:='PRED      '; SPPRED :=HASH;
09BC --      5324            IDREG:='WRITELN   '; SPWRITELN:=HASH; IDREG:='WRITE     '; SPWRITE:=HASH;
0A10 --      5325            IDREG:='READLN    '; SPREADLN :=HASH; IDREG:='READ      '; SPREAD :=HASH;
0A64 --      5326
0A64 --      5327            NEW(UNDFZERO,UNDEF); UNDFZERO@.UNDFIDNR:=0;
0AA2 --      5328            NEW(NILTYP,PTRTYP); NILTYP@.PTRIDNR:=0; NILTYP@.REFERTYP:=UNDFZERO;
0B1E --      5329            FORWTYPS:=NIL;
0B2A --      5330
0B2A --      5331            (* SET UP ACTIVE RECORDS OF PRE-DEFINED IDENTIFIERS *)
0B2A --      5332            (* I N T E G E R *)
0B2A --      5333            NEW(PTRACT,TYPID); NEW(PTRACT@.TYP,SCALTYP);
0B6C 1-      5334            WITH PTRACT@,TYP@ DO BEGIN
0B98 --      5335               IDNR:=SPINTEGER; LEVNR:=0; XREF:=NIL;
0BC4 -1      5336               SCIDNR:=SPINTEGER; SCLEVNR:=0; LOWBND:=-MAXINT; HIGBND:=MAXINT END;
0BFA --      5337            PUSH(PTRACT); TYPINTEGER:=PTRACT@.TYP;
0C54 --      5338            (* R E A L *)
0C54 --      5339            NEW(PTRACT,TYPID); NEW(PTRACT@.TYP,UNDEF);
0C98 1-      5340            WITH PTRACT@,TYP@ DO BEGIN
0CC4 -1      5341               IDNR:=SPREAL; LEVNR:=0; XREF:=NIL; UNDFIDNR:=SPREAL END;
0D10 --      5342            PUSH(PTRACT);
0D32 --      5343            (* B O O L E A N *)
0D32 --      5344            NEW(PTRACT,TYPID); NEW(PTRACT@.TYP,SCALTYP);
0D74 1-      5345            WITH PTRACT@,TYP@ DO BEGIN
0DA0 --      5346               IDNR:=SPBOOLEAN; LEVNR:=0; XREF:=NIL;
0DCC -1      5347               SCIDNR:=SPBOOLEAN; SCLEVNR:=0; LOWBND:=0; HIGBND:=1 END;
0E00 --      5348            PUSH(PTRACT); TYPBOOLEAN:=PTRACT@.TYP;
0E5A --      5349            (* T R U E *)
0E5A --      5350            NEW(PTRACT,SCALID);
0E70 1-      5351            WITH PTRACT@ DO BEGIN IDNR:=SPTRUE; LEVNR:=0; XREF:=NIL;
0EB2 -1      5352               STYP:=TYPBOOLEAN; VALUU:=0 END; PUSH(PTRACT);
0EFE --      5353            (* F A L S E *)
0EFE --      5354            NEW(PTRACT,SCALID);
0F14 1-      5355            WITH PTRACT@ DO BEGIN IDNR:=SPFALSE; LEVNR:=0; XREF:=NIL;
0F56 -1      5356               STYP:=TYPBOOLEAN; VALUU:=1 END; PUSH(PTRACT);
```

```
0FA4 --      5357            (* C H A R *)
0FA4 --      5358            NEW(PTRACT,TYPID); NEW(PTRACT@.TYP,SCALTYP);
0FE6 1-      5359            WITH PTRACT@,TYP@ DO BEGIN
1012 --      5360               IDNR:=SPCHAR; LEVNR:=0; XREF:=NIL;
103E -1      5361               SCIDNR:=SPCHAR; SCLEVNR:=0; LOWBND:=LOWORD; HIGBND:=HIGORD END;
1072 --      5362            PUSH(PTRACT); TYPCHAR:=PTRACT@.TYP;
10CC --      5363            (* O R D *)
10CC --      5364            NEW(PTRACT,FUNCID); NEW(PTRACT@.ARG); NEW(PTRACT@.ARG@.ARGTYP,SCALTYP);
1146 1-      5365            WITH PTRACT@,ARG@,ARGTYP@ DO BEGIN
118A --      5366               IDNR:=SPORD; LEVNR:=0; XREF:=NIL; PASSKND:=FALSE; NXTARG:=NIL;
11C2 --      5367               RETTYP:=TYPINTEGER; SWFORW:=FALSE;
11EC -1      5368               SCIDNR:=0; SCLEVNR:=0; LOWBND:=-MAXINT; HIGBND:=MAXINT END;
1218 --      5369               PUSH(PTRACT);
123A --      5370            (* P R E D *)
123A --      5371            NEW(PTRACT,FUNCID); NEW(PTRACT@.ARG);
1276 --      5372            NEW(PTRACT@.ARG@.ARGTYP,SCALTYP); NEW(PTRACT@.RETTYP,SCALTYP);
12DE 1-      5373            WITH PTRACT@,ARG@,ARGTYP@ DO BEGIN SWFORW:=FALSE;
1328 --      5374               IDNR:=SPPRED; LEVNR:=0; XREF:=NIL; PASSKND:=FALSE; NXTARG:=NIL;
1360 --      5375               RETTYP@.SCIDNR:=0; RETTYP@.SCLEVNR:=0;
1394 --      5376               RETTYP@.LOWBND:=-MAXINT; RETTYP@.HIGBND:=MAXINT;
13CC -1      5377               SCIDNR:=0; SCLEVNR:=0; LOWBND:=-MAXINT; HIGBND:=MAXINT END;
13F8 --      5378               PUSH(PTRACT);
141A --      5379            (* S U C C *)
141A --      5380            NEW(PTRACT,FUNCID); NEW(PTRACT@.ARG);
1456 --      5381            NEW(PTRACT@.ARG@.ARGTYP,SCALTYP); NEW(PTRACT@.RETTYP,SCALTYP);
14BE 1-      5382            WITH PTRACT@,ARG@,ARGTYP@ DO BEGIN SWFORW:=FALSE;
1508 --      5383               IDNR:=SPSUCC; LEVNR:=0; XREF:=NIL; PASSKND:=FALSE; NXTARG:=NIL;
1540 --      5384               RETTYP@.SCIDNR:=0; RETTYP@.SCLEVNR:=0;
1574 --      5385               RETTYP@.LOWBND:=-MAXINT; RETTYP@.HIGBND:=MAXINT;
15AC -1      5386               SCIDNR:=0; SCLEVNR:=0; LOWBND:=-MAXINT; HIGBND:=MAXINT END;
15D8 --      5387               PUSH(PTRACT);
15FA --      5388            (* C H R *)
15FA --      5389            NEW(PTRACT,FUNCID); NEW(PTRACT@.ARG);
1636 1-      5390            WITH PTRACT@,ARG@ DO BEGIN
1662 --      5391               IDNR:=SPCHR; LEVNR:=0; XREF:=NIL; PASSKND:=FALSE; NXTARG:=NIL;
169A --      5392               RETTYP:=TYPCHAR; SWFORW:=FALSE;
16C4 -1      5393               ARGTYP:=TYPINTEGER END; PUSH(PTRACT);
170A --      5394
170A --      5395            IMPLTYPNR:=0; COMPNEST:=0;
1722 --      5396            (* SET UP A SENTINEL (DUMMY HEADER) FOR TOPBOOL *)
1722 --      5397            NEW(TOPBOOL); TOPBOOL@.UP:=NIL; TOPBOOL@.DOWN:=NIL;
1778 --      5398            (* INITIALIZE COMPILER OPTIONS TO $S+,X+ *)
1778 --      5399            NOTSTAND:=TRUE; XREFLIST:=TRUE;
1794 --      5400
1794 --      5401            (* INITIALIZATION OF STRING SYSTEM *)
1794 --      5402            REMWS:=1; ALLOCS:=NIL; EXTRAS:=NIL;
17BA --      5403
17BA --      5404            LOC(RETRIEVE, 10); LIT(RETRIEVE, 10, '$RETRIEVE 12345');
17EC --      5405            LOC(REPLACE, 9); LIT(REPLACE, 9, '$REPLACE 012345');
181E --      5406            LOC(DOLLAR, 1); LIT(DOLLAR, 1, '$23456789012345');
1850 --      5407            LOC(LITCOMMA, 1); LIT(LITCOMMA, 1, ',23456789012345');
1882 --      5408            LOC(LITCOLON, 1); LIT(LITCOLON, 1, ':23456789012345');
18B4 --      5409            LOC(LITBLANK, 1); LIT(LITBLANK, 1, ' 23456789012345');
18E6 --      5410            LOC(PI, 4); LIT(PI, 4, '$PI$56789012345');
1918 --      5411            LOC(STM, 4); LIT(STM, 4, '$STM56789012345');
194A --      5412            LOC(OMEGA, 6); LIT(OMEGA, 6, '$OMEGA789012345');
197C --      5413            LOC(EQUIV, 1); LIT(EQUIV, 1, '=23456789012345');
19AE --      5414            LOC(TERMIN, 1); LIT(TERMIN, 1, '.23456789012345');
```

```
19E0 --    5415            LOC(VALPREF, 5); LIT(VALPREF, 5, '$VAL$6789012345');
1A12 --    5416            LOC(DUMMYCOD, 50); DUMMYSBND:=NIL;
1A34 --    5417            (* INITIALIZE COMPILER OPTION TO $U- *)
1A34 --    5418            SUPERSCR:=FALSE;
1A40 -0 A  5419         END;
1F2C --    5420
1F2C --    5421      (* MAIN PROGRAM STARTS HERE *)
0000 0-    5422      BEGIN
0066 --    5423         HEADER:=' STMNR LEV NST SEMIC   SOURCE CODE:          ';
006C --    5424         PAGENR:=0; PROGID:=0; NEWPAGE;
007C --    5425         REWRITE(SPUNCH);
0086 --    5426         REWRITE(ERRORS);
0090 --    5427         (*DE-0 REWRITE(DEBUG); 0-BUG*)
0090 --    5428         INIT; (* INITIALIZE *)
0094 --    5429         ADVANCE;
0098 --    5430         PROGRAMM;
009C --    5431
009C --    5432      9999: (* EPILOG FOR COMPILATION *)
00A6 --    5433         SWERRMSG:=FALSE;
00B2 --    5434         WHILE NOT EOF(INPUT) DO NEXTCH; SWERRMSG:=TRUE;
00DA --    5435         LEVEL:=0; POP;
00EA --    5436         (* X-REF WANTED? *)
00EA 1-    5437         IF XREFLIST THEN BEGIN
00FA --    5438            HEADER:='FIRSTR IDENTIFIER CLASS, TYPE, REFERENCES:   ';
0100 --    5439            NEWPAGE;
0104 --    5440            WHILE ATTR<>NIL DO
0116 2-    5441               BEGIN PRINTXREF(ATTR);
013E --    5442                  (* PRINT ONLY IF FIRST CHARACTER <> EOSTRGCH *)
013E 3-    5443                  IF ORD(XRFF@)<>ORDEOSTRGCH THEN BEGIN WRITE(OUTPUT,' ');
016C --    5444                  FOR I:=1 TO 18 DO
0172 44    5445                     BEGIN OUTPUT@:=XRFF@; PUT(OUTPUT); GET(XRFF) END;
01CE --    5446                  (* READ ONE EXTRA BLANK AFTER EOLN *)
01CE --    5447                  GET(XRFF);
01D8 --    5448                  WHILE ORD(XRFF@)<>ORDEOSTRGCH DO
01F2 4-    5449                     BEGIN FOR I:=19 TO MAXLNL+23 DO
020A --    5450                              IF ORD(XRFF@)<>ORDEOSTRGCH THEN
023C 5-    5451                                 BEGIN OUTPUT@:=XRFF@; GET(XRFF);
0264 --    5452                                    IF OUTPUT@=' ' THEN WHILE XRFF@=' ' DO GET(XRFF);
02A4 --    5453                                    PUT(OUTPUT)
02AC -5    5454                                 END;
02CC --    5455                           WRITELN(OUTPUT,' '); LINECNT;
02E8 --    5456                           WRITE(OUTPUT,'                    ')
02FC -4    5457                     END;
0300 -3    5458                  WRITELN(OUTPUT,' '); LINECNT; END; (* OF PRINTOUT *)
031C --    5459                  ATTR:=ATTR@.NXTACT
0336 -2    5460               END
035A -1    5461            END;
035E --    5462         HEADER:='ERRNR SEMIC COL ERROR MESSAGE LISTING:       ';
0364 --    5463         NEWPAGE;
0368 --    5464         RESET(ERRORS);
0372 --    5465         WHILE NOT EOF(ERRORS) DO
038A 1-    5466            BEGIN OUTPUT@:=ERRORS@; PUT(OUTPUT);
03B0 22    5467               IF EOLN(ERRORS) THEN BEGIN WRITELN(OUTPUT,' '); LINECNT END;
03E4 -1    5468               GET(ERRORS) END;
03F2 --    5469         IF ERRCNT=0 THEN WRITELN(OUTPUT,'***NO ERRORS DETECTED IN THIS PROGRAM!***')
0418 -0    5470      END .

*AAEC PASCAL COMPILATION CONCLUDED *
```

*NO ERRORS DETECTED IN PASCAL PROGRAM *

IDENT	CLASS	DEFINITION	REFERENCES
ABS	FUNCTION	STANDARD	337 887
ACT	VARIABLE	1142	1146
ACTEXPLS	VARIABLE	3381	3401 3406 3408
ACTEXPLS	VARIABLE	3415	3486 3489 3489 3493 3496 3501 3515 3516 3518 3522 3522 3524
ACTEXPLS	VARIABLE	4180	4296 4303 4373 4374 4378 4380 4383 4392 4394 4398 4400 4415 4416 4418 4423 4424 4426 4426 4428
ACTION	CONSTANT	476	516
ACTIVS	TYPE	105	113 121 132 164 264 266 1052 1054 1084 1111 1142 1187 1218 1221 1223 1243 1245 1299 1300 1344 1522 1567 1598 1651 1766 1857 1941 2028 2237 2273 2273 2275 2314 2316 2364 2397 2399 2444 2491 2492 2588 2646 2734 2819 2824 2922 2923 2926 2968 2970 3006 3222 3415 4176 4177 4706 4706 4767 4773 4775 4978 5017 5017 5170 5233
ACTLST	VARIABLE	2968	2987 2994
ACTLST	VARIABLE	3006	3025 3033 3070 3094 3098 3107 3108 3109 3109
ACTLST	TYPE	164	164 258 1188 1218 1526
ACTLSTEL	VARIABLE	1188	1193 1194 1195 1196
ACTLSTPT	FIELD	164	1090 1093 1194 1331 1545 3090 3709
ACTPARAM	PROCEDUR	3381	3486 4296 4303 4373 4392
ADVANCE	PROCEDUR	475	999 1012 1027 1676 1713 1722 1748 1791 1794 1797 1822 1831 1834 1845 1879 1909 1963 1999 2082 2130 2175 2204 2290 2335 2380 2414 2471 2508 2530 2565 2573 2608 2615 2669 2704 2765 2799 2840 2850 2863 2873 2941 2984 3022 3030 3062 3188 3242 3322 3355 3395 3554 3606 3670 3683 3698 3705 3708 3713 3721 3727 3810 3919 3933 3980 4058 4084 4492 4500 4532 4599 4607 4650 4657 4684 4691 4721 4750 4794 4799 4812 4847 4861 4893 4904 4923 4956 4999 5182 5429
ALLOCS	VARIABLE	301	757 757 764 780 838 859 874 5402
ALLOWFW	VARIABLE	290	2619 5035 5048
AMPERSAN	CONSTANT	87	638 5268
ANDSYM	CONSTANT	71	482 484 583 587
ANYTIME	VARIABLE	1245	1253 1255 1260
APP	PROCEDUR	908	941 942 1513 1514 1515 1517 1553 1555 1578 1579 1585 1586 1587 1592 1602 1604 1613 1622 1639 1640 3113 3115 3118 3121 3125 3126 3127 3270 3274 3283 3309 3311 3495 3496 3808 3815 3917 3924 3931 3938 3985 4056 4063 4082 4089 4228 4230 4232 4237 4239 4241 4243 4261 4269 4278 4284 4317 4325 4335 4342 4346 4349 4360 4361 4366 4414 4423 4443 4449 4453 4460 4827 4830 4838 4841 4876 4880 4884 4887 5080 5082 5089 5090 5090 5104 5112 5116 5117 5117 5124 5125 5125 5132 5133 5133 5146 5155 5219 5221
APPENDIX	VARIABLE	1598	1600 1601 1602 1612 1613 1620 1621 1621 1622 1628
APPENDIX	VARIABLE	4771	4862 4878 4878 4878 4880 4880 4882 4882 4883 4884 4884
APPENV	PROCEDUR	1596	4357 4840 4886
ARG	FIELD	118	1161 1323 3070 3084 3469 3484 3512 4390 4411 5364 5364 5365 5371 5372 5373 5380 5381 5382 5389 5390
ARGLST	VARIABLE	2968	2987 2994
ARGLST	VARIABLE	3006	3025 3033 3070 3084
ARGLST	VARIABLE	3415	3482 3484 3504 3505 3512 3515 3517 3519 3523 3523 3524
ARGLST	VARIABLE	4180	4389 4390 4398 4403 4404 4411 4415 4417 4419 4427 4427 4428
ARGS	TYPE	157	160 118 1144 2819 2825 2922 2923 2925 2968 2970 3006 3415 3416 4180
ARGTYP	FIELD	158	1164 2907 3507 3517 4406 4417 5364 5365 5372 5373 5381 5382 5393
ARRAYSYM	CONSTANT	71	2503 2506 2599 2611
ARRTYP	CONSTANT	102	129 1069 1119 1254 1359 1393 1824 2212 2521 3047 3255 5092 5097 5107
ATTR	VARIABLE	266	1229 1237 1240 5311 5440 5441 5459 5459
AUXIL	VARIABLE	910	918 919 920
AUXIL	VARIABLE	933	937 937 938 940 942 942
AUXIL	VARIABLE	969	979 980 981
BECOMES	CONSTANT	81	659 3235 3365 4192 4196 4213 4917
BEGINSYM	CONSTANT	71	1743 1754 1954 1974 1994 2016 2659 2679 2699 2719 2760 2774 2794 2809 3150 3159 4787 4797 4989 5003 5031 5052 5310
BEYOND	VARIABLE	752	773 785 787 787 798 800 800
BLANK	CONSTANT	88	693 5271
BLDLAB	FUNCTION	1052	1063 1716 1751
BLDSTRG	FUNCTION	1066	1079 1842 3697
BLKLEVNR	VARIABLE	5019	5065 5066 5071 5139
BLOCK	PROCEDUR	1657	3129 5015 5214
BND	FIELD	194	3302 3303 4230 4239 4246
BODYNR	VARIABLE	5019	5150 5151 5153 5159

```
IDENT     CLASS     DEFINITION   REFERENCES
BOOLEAN   TYPE       STANDARD    120 131 133 134 146 159 178 205 234 255 290 293 294 320 336 413 479 483 1082 1084 1223 1245 1343
                                 1402 1409 1894 2187 2489 2824 3007 3169 3172 3213 3222 3383 4584 5018
BOT       VARIABLE    483        484 485 486 488
C         VARIABLE   5233        5259 5259 5260 5260 5261 5261
CARACTLS  VARIABLE   2923        2959 2960
CARARGLS  VARIABLE   2923        2956 2957
CARCONSL  VARIABLE   1892        1927 1928 1929 1930 1934 1936
CARCONSL  VARIABLE   4581        4626
CAREXPLS  VARIABLE   3170        3200 3201 3202 3203 3204 3208 3210
CARIDLST  VARIABLE   1659        1684 1684 1685 1689 1691
CARRECLS  VARIABLE   2273        2305 2306
CARSMPLS  VARIABLE   2187        2212 2212 2218 2220 2225 2227 2228 2232 2234
CASCONSL  VARIABLE   4769        4857
CASEELEM  PROCEDUR   4547        4600 4856
CASELEML  PROCEDUR   4581        4601 4857
CASESYM   CONSTANT     71        2454 2468 4787 4845 4988 5003 5310
CAT       PROCEDUR    931        1621 1638 1641 3111 3120 3278 3307 3449 3678 3694 4277 4334 4347 4364 4504 4505 4806 4829 4836 4873
                                 4881 4882 5131 5154 5157
CDRACTLS  VARIABLE   2926        2943 2947 2959 2960
CDRARGLS  VARIABLE   2925        2943 2947 2956 2957
CDRCONSL  VARIABLE   1894        1911 1915 1927 1930
CDRCONSL  VARIABLE   4584        4601 4606 4611 4620 4626
CDREXPLS  VARIABLE   3173        3194 3198 3204
CDRIDLST  VARIABLE   1661        1678 1682 1685
CDRRECLS  VARIABLE   2275        2292 2298 2305 2306
CDRRECLS  VARIABLE   2444        2461 2465 2466 2476 2478 2479
CDRSMPLS  VARIABLE   2189        2206 2210 2228
CH        VARIABLE    237        444 445 457 462 469 498 510 514 517 525 528 530 533 555 556 564 565 566 566 570 570 571 606 608 615
                                 617 617 617 618 620 620 623 624 625 630 631 632 646 647 653 659 672 674 681 693 698 705 5237
CHAR      TYPE       STANDARD    203 210 213 215 216 217 219 222 226 230 232 237 267 299 305 326 333 355 369 392 397 477 946 966 966
                                 1032 1036 1219 1343 1450 5233
CHARVAR   TYPE        189        193 194 306 311 312 313 314 315 316 317 318 728 746 852 884 890 890 908 908 931 931 945 964 966
                                 1448 1481 1522 1527 1559 1567 1596 1598 1631 1634 1653 3008 3169 3172 3216 3217 3219 3220 3383 3411
                                 3413 3414 3651 3654 3772 3773 3775 3835 3837 3892 3893 3895 3958 3960 4033 4034 4036 4109 4179 4474
                                 4476 4764 4766 4771 5020 5171
CHR       FUNCTION  STANDARD    445 533 676 1184 1509 5237 5259 5259 5272 5273 5274
CHRSEL    VARIABLE    219        498 510 565 566 566 570 570 571 5259 5260 5261 5262 5262 5263 5263 5264 5264 5265 5265 5266 5266
                                 5267 5267 5268 5268 5269 5269 5270 5270 5271 5271 5272 5273
CH1       VARIABLE    966        972 985
CH2       VARIABLE    966        972 986
COD       VARIABLE   1596        1602 1603 1603 1604 1613 1614 1622 1623
CODE      VARIABLE    369        379
CODIDENT  PROCEDUR   1522        1577 1584 1612 1620 3106 3110 3464 3494 4224 4268 4276 4324 4333 4413 4442 4452 5078 5130
CODIN     VARIABLE   3216        3311 3312 3318 3351 3362 3369
CODIN     VARIABLE   3413        3418 3446 3449 3449 3450 3455 3456 3457 3464 3476 3535
CODINTEG  PROCEDUR   1448        1506 1554 1637 3119 3280 3286 3446 3457 3691 3704 4503 4829 4835 4878 4881 5103 5110 5217
CODOUT    VARIABLE   3217        3318 3351 3362 3369
CODOUT    VARIABLE   3411        3476 3489 3494 3495 3496 3497
COL       FUNCTION    497        498 499 500 516 549
COLONCHR  CONSTANT     88        657 5270
COLONSYM  CONSTANT     81        660 1671 1680 1904 1913 2257 2326 2329 2416 2767 2802 2842 2853 2866 2979 2990 3034 4567 5000
COMMACHR  CONSTANT     87        638 5267
COMMASYM  CONSTANT     80        1671 1674 1708 1711 1904 1907 2075 2097 2199 2202 3183 3186 3235 3364 3427 3433 3549 3557 3601 3604
                                 3788 3818 3906 3941 4048 4091 4716 4719
COMPACT   PROCEDUR    746        881
COMPARE   PROCEDUR   1341        1360 1373 1382 1428 3260 3516 4215 4416 4685 4692 4918
COMPNEST  VARIABLE    288        1380 1381 1381 1383 1383 5395
COMPON    VARIABLE   5017        5096 5097 5099 5099 5106 5107 5109 5110 5111 5114 5114
```

IDENT	CLASS	DEFINITION	REFERENCES
COMPONEN	VARIABLE	2491	2514 2515 2525
COMPOPND	FUNCTION	1402	1411 1441 3812 3921 3935 3982 4060 4086
COMPSTMC	VARIABLE	4474	4495 4505 4505 4506
COMPSTMC	VARIABLE	5020	5063 5067 5079 5080 5081 5082 5143 5146 5147 5149 5153 5154 5154 5155 5157 5157 5158 5159 5160
COMPSTMR	PROCEDUR	4474	4495 4804 5153
COMPTYP	FIELD	130	1077 1123 1255 1360 1360 2228 2519 2525 3314 5099 5114
COMTYPID	VARIABLE	2824	2844 2846 2855 2857 2867 2867 2876 2881 2883 2885 2886 2889
CONDCOD	VARIABLE	4766	4813 4815 4827 4827 4863 4865 4876 4876
CONDTYP	VARIABLE	4766	4815 4816 4817 4865 4866 4867 4896 4897 4898
CONSDEFP	PROCEDUR	1939	5034
CONSDFPT	PROCEDUR	1943	1972 2014
CONSLST	VARIABLE	2239	2256
CONSLST	VARIABLE	4547	4566 4572
CONSTANT	PROCEDUR	1855	1910 1968 2010 2083 2170 2255 4565
CONSTLIS	PROCEDUR	1891	1911 2256 4566
CONSTSYM	CONSTANT	71	1742 1754 1994 1997 5030
CONSTYP	VARIABLE	1764	1785 1788 1789 1792 1795 1796 1802 1803 1824 1829 1832 1833 1838 1842 1850 1851
CONSTYP	VARIABLE	1855	1873 1876 1877 1881 1886 1887
CONSTYP	VARIABLE	1894	1910 1911
CONSTYP	VARIABLE	1941	1966 1967 1968 1969 2008 2009 2010 2011
CONSTYP	VARIABLE	2239	2255 2256
CONSTYP	VARIABLE	4549	4565 4566
CONSTYP1	VARIABLE	2029	2035 2037 2042 2043 2054 2055 2090 2168
CONSTYP2	VARIABLE	2029	2036 2038 2042 2043 2083 2170
CONSVAL	VARIABLE	1764	1786 1790 1793 1796 1803 1827 1827 1830 1833 1839 1843 1851
CONSVAL	VARIABLE	1855	1874 1878 1881 1887
CONSVAL	VARIABLE	1894	1910 1911
CONSVAL	VARIABLE	1941	1967 1968 1969 2009 2010 2011
CONSVAL	VARIABLE	2239	2255 2256
CONSVAL	VARIABLE	4549	4565 4566
CONSVAL1	VARIABLE	2029	2048 2056 2091 2168
CONSVAL2	VARIABLE	2029	2048 2056 2083 2170
CONTRTYP	VARIABLE	4667	4685 4692
CONTRTYP	VARIABLE	4773	4910 4914 4915 4916 4918 4919
CONV	FUNCTION	333	351 1038 1452
CONVPSID	FUNCTION	1031	1047 1049 1715 1750 2151 4958 4995
COUNT	VARIABLE	1526	1534 1535 1535 1536 1536 1537 1537 1539 1542 1546 1546 1551
CURACT	VARIABLE	2926	2942 2943
CURACT	VARIABLE	2970	2985 2987
CURARG	VARIABLE	2925	2942 2943
CURARG	VARIABLE	2970	2985 2987
CURCASE	VARIABLE	480	510 511 596 599 639
CURCONSL	VARIABLE	4584	4600 4601
CUREXPCO	VARIABLE	3172	3189 3190 3194
CUREXPTY	VARIABLE	3172	3190 3191 3194
CURIDNR	VARIABLE	1661	1677 1678
CURLG	FIELD	186	733 736 786 787 788 794 795 800 801 803 842 867 878 893 900 902 902 903 904 912 912 918 918 922 924 925 927 927 927 937 958 973 979 983 986 987 987 1469 1483 1488 1508 1539 1603 3105 3112 3117 3122 3122 4356 4448 4451 5067 5068 5068 5079 5143
CURNAME	VARIABLE	1567	1569 1577 1579 1584 1587 1593
CUROPCOD	VARIABLE	3775	3779 3811 3815 3832
CUROPCOD	VARIABLE	3837	3839 3851 3852 3889
CUROPCOD	VARIABLE	3895	3897 3920 3924 3934 3938 3955
CUROPCOD	VARIABLE	3960	3962 3981 3985 3990 3991 4030
CUROPCOD	VARIABLE	4109	4111 4129 4130 4170
CUROPTYP	VARIABLE	3776	3811 3812 3813
CUROPTYP	VARIABLE	3837	3851 3852
CUROPTYP	VARIABLE	3895	3920 3921 3922 3934 3935 3936

IDENT	CLASS	DEFINITION	REFERENCES
EXPCOD	VARIABLE	1653	4130
EXPCOD	VARIABLE	4179	4182 4214 4219 4221 4470
EXPLST	TYPE	143	147 3170 3173 3219 3381 3415 3416 4180 4180
EXPLSTCO	FIELD	144	3201 3262 3489 3496 4423 4424
EXPLSTTY	FIELD	145	3202 3261 3507 3516 4378 4406 4416
EXPRESLI	PROCEDUR	3169	3194 3245 3401
EXPRESSI	PROCEDUR	1653	3190 3244 3397 3555 3581 3728 4107 4214 4685 4692 4815 4848 4865 4896 4918
EXPTYP	VARIABLE	1653	4130 4136 4138
EXPTYP	VARIABLE	4176	4214 4215
EXPTYP	VARIABLE	4669	4685 4685 4692 4692
EXTENSIO	VARIABLE	3220	3224 3288 3303 3309 3378
EXTRAS	VARIABLE	302	770 770 834 835 835 835 870 871 871 871 5402
FACCOD	VARIABLE	3651	3671 3678 3678 3679 3684 3692 3694 3694 3695 3704 3715 3728
FACTOR	PROCEDUR	3651	3671 3811 3851
FACTORLI	PROCEDUR	3772	3816 3852
FACTYP	VARIABLE	3651	3671 3672 3673 3676 3684 3690 3697 3703 3709 3714 3722 3728 3729 3737
FALSE	CONSTANT	STANDARD	446 491 569 668 1023 1088 1103 1107 1234 1253 1326 1346 1411 1412 1689 1760 1802 1850 1886 1934 1980 2022 2110 2182 2232 2269 2360 2438 2582 2612 2639 2685 2725 2780 2815 2839 2860 2870 2918 3022 3083 3097 3135 3165 3208 3375 3406 3441 3467 3475 3479 3508 3532 3563 3586 3616 3647 3669 3687 3702 3708 3712 3718 3737 3795 3829 3856 3913 3929 3952 3976 3997 4055 4067 4101 4136 4407 4468 4543 4577 4616 4700 4755 4972 5011 5048 5140 5164 5235 5366 5367 5373 5374 5382 5383 5391 5392 5418 5433
FIELDLIS	PROCEDUR	1651	2259 2381 2442 2531
FIELDLST	PROCEDUR	2364	2461
FILELIST	VARIABLE	5170	5189 5191 5192 5200 5210 5210
FILESYM	CONSTANT	73	2503 2562 2599 2611
FIRCONSL	VARIABLE	4581	4610 4613 4619 4620 4623 4624 4624
FIRCONSL	VARIABLE	4769	4856 4857
FIRSTACT	VARIABLE	2922	2949 2959
FIRSTARG	VARIABLE	2922	2949 2956
FIRSTCOD	VARIABLE	3169	3201
FIRSTCOD	VARIABLE	3219	3243 3244 3245
FIRSTCOD	VARIABLE	3383	3396 3397 3401
FIRSTFIL	VARIABLE	5170	5188 5189
FIRSTID	VARIABLE	1659	1684
FIRSTID	VARIABLE	2064	2081 2087 2099 2101
FIRSTID	VARIABLE	2314	2332 2340
FIRSTID	VARIABLE	2399	2413 2415
FIRSTID	VARIABLE	2445	2472 2473
FIRSTID	VARIABLE	2731	2764 2766 2800 2801
FIRSTID	VARIABLE	2823	2839 2841 2851 2852 2864 2865 2874 2875
FIRSTSTM	VARIABLE	4474	4503
FIRSTSTM	VARIABLE	4764	4800 4804 4808
FIRSTSWV	VARIABLE	3169	3203
FIRSTSWV	VARIABLE	3383	3399 3400 3401
FIRSTTYP	VARIABLE	1891	1917 1919 1920 1924
FIRSTTYP	VARIABLE	2187	2213 2214 2220 2222 2227
FIRSTTYP	VARIABLE	3169	3202
FIRSTTYP	VARIABLE	3219	3244 3245
FIRSTTYP	VARIABLE	3383	3397 3398 3401
FIRSTVAL	VARIABLE	1891	1929
FIRSTVAR	VARIABLE	2273	2301 2305
FLDID	CONSTANT	101	117 1159 1252 1263 1310 1310 1322 1576 1610 2331 2421 3462 4202 4266 4322 4438 4725 4740 4927 4942 5076
FLDIDLST	VARIABLE	2399	2415 2420 2423 2428 2428
FLDLST	VARIABLE	1651	2460 2462 2466 2473 2477 2478 2479
FLDLST	VARIABLE	2491	2531 2531 2533
FLDS	VARIABLE	1245	1261 1262 1264 1265 1266 1267 1267
FLDTYP	VARIABLE	2399	2418 2419 2425

IDENT	CLASS	DEFINITION	REFERENCES
FORMALPA	PROCEDUR	2819	2942 2985
FORMPARL	PROCEDUR	2922	2943 2987
FORMPARP	PROCEDUR	2968	3025 3033
FORSTMTR	PROCEDUR	4667	4919
FORSYM	CONSTANT	73	4788 4902 4989 5004
FORWARGS	FIELD	121	1326 3070 3098
FORWELEM	VARIABLE	2588	2621 2621 2624 2625 2626 2626
FORWLST	TYPE	170	172 286 2588
FORWSW	VARIABLE	3007	3027 3058 3061
FORWTYPS	VARIABLE	286	2626 2626 5037 5039 5040 5042 5044 5045 5047 5047 5329
FREE	PROCEDUR	884	899 920 942 955 981 1459 1518 1533 1555 1593 1628 3124 3126 3128 3311 3378 3378 3378 3450 3535 3678 3694 3832 3832 3889 3955 3955 4030 4104 4170 4246 4246 4424 4470 4470 4470 4470 4470 4471 4512 4806 4808 4827 4838 4841 4843 4876 4882 4884 4887 4889 4966 5142 5142 5142 5154 5155 5157 5160 5221
FREESPAC	VARIABLE	228	421 431 431 431 5306
FROMTYP	VARIABLE	4773	4918 4918
FUNCID	CONSTANT	101	118 1095 1160 1168 1311 1311 1323 1329 1545 1581 1616 3064 3078 3091 3466 3480 3481 4203 4272 4328 4437 5075 5364 5371 5380 5389
FUNCSW	VARIABLE	3007	3022 3030 3064 3071 3078 3085
FUNCSYM	CONSTANT	73	1742 1754 1954 1974 1994 2016 2659 2679 2699 2719 2760 2774 2794 2809 2835 2859 3018 3029 3150 3153 5030 5310
FUNCTYP	VARIABLE	3006	3035 3036 3040 3043 3044 3047 3055 3071 3085
GET	PROCEDUR	STANDARD	5445 5447 5451 5452 5468
GOTOSYM	CONSTANT	73	4788 4954 4990 5004
GREATRCH	CONSTANT	88	651 5269
HASH	FUNCTION	412	433 582 1047 5314 5314 5315 5315 5316 5316 5317 5317 5318 5318 5319 5319 5320 5320 5321 5321 5322 5322 5323 5323 5324 5324 5325 5325
HASHNAME	VARIABLE	226	357 408 417 417 419 425 1097 1115 1130 1133 1136 1154 1173 1233 1233 1287 1289 1293 1305 1535 1537 2541 3051 3338 5185 5305
HASHNEXT	VARIABLE	227	418 431
HASHPTR	TYPE	96	106 126 136 138 155 211 226 227 227 228 258 278 406 412 413 1032 1052 1082 1299 1526 1649 1659 1696 1941 2026 2028 2064 2314 2316 2399 2445 2489 2588 2646 2731 2823 2824 3007 3222 3411 3653 4173 4705 4762 4777 4978 5170
HEADER	VARIABLE	210	359 5423 5438 5462
HELPER	VARIABLE	3008	3103 3110 3111 3111 3115 3119 3120 3120 3121 3122 3124
HELPER	VARIABLE	4179	4182 4227 4228 4229 4230 4231 4232 4236 4237 4238 4239 4240 4241 4242 4243 4268 4269 4276 4277 4277 4278 4324 4325 4333 4334 4334 4335 4341 4342 4344 4346 4347 4358 4360 4361 4363 4364 4442 4443 4452 4453 4470
HELPER	VARIABLE	4764	4805 4805 4806 4806 4813 4829 4829 4829 4830 4833 4835 4836 4836 4838 4838 4862 4879 4880 4881 4881 4881 4882 4882
HELPER	VARIABLE	5020	5059 5078 5082 5103 5104 5111 5112 5130 5131 5131 5132 5145 5146 5156 5157 5157
HELPER	VARIABLE	5171	5216 5217 5219 5220 5221 5221
HIGBND	FIELD	128	1060 1076 1116 1117 1356 1357 1367 1367 2056 2155 2163 3286 5110 5336 5347 5361 5368 5376 5377 5385 5386
HIGORD	CONSTANT	65	5259 5361
I	VARIABLE	336	339 340 346 347 347
I	VARIABLE	438	442 442
I	VARIABLE	477	563 564 564 564 565 567
I	VARIABLE	1450	1467 1468 1468
I	VARIABLE	5233	5236 5236 5274 5274 5274 5304 5305 5305
I	VARIABLE	325	400 400 400 402 402 402 408 408 408 735 795 796 904 905 905 924 925 926 960 961 961 983 984 984 1017 1017 1017 1019 1019 1019 1040 1040 1040 1043 1043 1043 1046 1046 1046 1099 1099 1099 1174 1174 1174 1289 1289 1289 1292 1292 1293 1306 1306 1306 1396 1396 1396 1508 1509 1509 2542 2542 2542 3052 3053 3053 3339 3340 3340 5444 5449
ID	VARIABLE	1082	1089 1090 1093 1097
ID	VARIABLE	1299	1302 1305 1313 1321 1331
ID	VARIABLE	1526	1529 1535 1537 1541
ID	VARIABLE	1941	2000 2003
ID	VARIABLE	2028	2131 2132 2174 2176

IDENT	CLASS	DEFINITION	REFERENCES
ID	VARIABLE	2588	2616 2616 2628 2632
ID	VARIABLE	2646	2705 2708 2713 2714 2715
ID	VARIABLE	3222	3323 3330 3332 3338
ID	VARIABLE	3411	3438 3461 3480 3482
ID	VARIABLE	3653	3682 3684
ID	VARIABLE	4173	4198 4200 4290 4290 4291 4291 4299 4299 4307 4353 4372 4387 4389
ID	VARIABLE	4706	4722 4724 4727
ID	VARIABLE	4762	4793 4795 4905 4907 4909 4924 4926 4929
ID	FIELD	155	1684 2141 2145 2423 2737 2740 2894 2897 5192 5200
IDCAT	VARIABLE	1299	1308 1319
IDCLASS	TYPE	101	114 1299
IDENTIFI	CONSTANT	78	215 482 494 581 1677 1779 1782 1867 1870 1954 1957 2000 2125 2131 2173 2249 2253 2336 2409 2412 2454 2457 2472 2598 2603 2616 2659 2662 2705 2760 2763 2800 2835 2838 2846 2851 2857 2864 2867 2874 3023 3027 3031 3038 3058 3323 3579 3632 3638 3664 3681 3849 3972 3987 4121 4559 4563 4722 4787 4792 4905 4924 4988 5003 5183 5188
IDENTLIS	PROCEDUR	1659	1678 2132 2415 2766 2801 2841 2852 2865 2875 5189
IDENTREM	PROCEDUR	3411	3684
IDKLASS	FIELD	114	1057 1095 1155 1168 1252 1259 1260 1286 1329 1545 1574 1608 1784 1872 1961 2004 2086 2100 2345 2629 2884 3042 3091 3439 3481 4199 4264 4320 4388 4395 4436 4725 4908 4927 5041 5074
IDLST	TYPE	155	155 1659 1661 2028 2030 2399 2731 2823 5170
IDNR	FIELD	106	1058 1130 1154 1173 1195 1196 1233 1233 1264 1264 1276 1281 1287 1289 1293 1313 1529 1960 1970 2003 2012 2145 2332 2423 2539 2539 2541 2666 2673 2674 2675 2708 2740 2897 3067 3081 3332 3488 3491 3491 3492 4741 4741 4943 4943 5121 5122 5195 5203 5335 5341 5346 5351 5355 5360 5366 5374 5383 5391
IDREG	VARIABLE	216	415 415 415 416 417 425 487 489 563 564 576 1039 1040 1046 5314 5314 5315 5315 5316 5316 5317 5317 5318 5318 5319 5319 5320 5320 5321 5321 5322 5322 5323 5323 5324 5324 5325 5325
IDVAL	VARIABLE	1031	1038
IFSYM	CONSTANT	73	4787 4810 4989 5003
IMPLTYPN	VARIABLE	285	2150 2150 2151 5395
INDXEXPL	VARIABLE	3219	3245 3253 3261 3262 3316 3316
INDXTYP	FIELD	129	1072 1073 1074 1075 1076 1122 1361 1362 1364 1364 1365 1365 1366 1366 1367 1367 2218 2220 2225 2227 2523 3260 3263 3265 3266 3281 3286 3287 5109 5110 5111
INDXTYPL	VARIABLE	2491	2511 2516 2517 2518 2519 2519
INIT	PROCEDUR	5232	5428
INPUT	VARIABLE	41	444 457 457 462 462 463 470 5434
INSERTWO	VARIABLE	217	401 402 1016 1017 1018 1019 1097 1099 1173 1174 1305 1306 2541 2542 3051 3053 3338 3340
INSYM	CONSTANT	74	583 584
INTEGER	TYPE	STANDARD	107 116 127 128 151 167 171 184 186 207 208 221 233 239 260 261 262 271 285 303 319 325 333 333 336 369 438 477 478 483 497 730 746 751 752 852 945 1002 1031 1036 1066 1402 1448 1450 1479 1526 1631 1661 1696 1764 1855 1891 1894 1941 2029 2030 2239 2731 3007 3776 4036 4474 4476 4515 4549 4764 4766 4771 4775 4777 4978 5019 5233
ITSFIELD	VARIABLE	1066	1076
J	VARIABLE	336	343 344 344 345 345 348 349 350 350 350
J	VARIABLE	370	379 380 381 382 383 383 383 383 384 384
J	VARIABLE	477	576 576 576
J	VARIABLE	1344	1391 1396
J	VARIABLE	1450	1454 1455 1458 1460 1469
K	VARIABLE	1450	1464 1466 1468
KEYW	VARIABLE	213	393 487 489 5239 5239 5239 5240 5240 5240 5241 5241 5241 5242 5242 5242 5243 5243 5243 5244 5244 5244 5245 5245 5245 5246 5246 5246 5247 5247 5247 5248 5248 5248 5249 5249 5249 5250 5250
LABELDCL	PROCEDUR	1694	5033
LABELSYM	CONSTANT	74	1742 1746 5030
LABLDCLR	PROCEDUR	1698	1717 1752
LABNR	VARIABLE	1052	1058
LABNR	VARIABLE	1696	1715 1716 1716 1750 1751 1751
LABNR	VARIABLE	4777	4958 4959
LABNR	VARIABLE	4978	4995 4996
LABVAL	VARIABLE	1696	1714 1715 1749 1750
LABVAL	VARIABLE	4777	4957 4958

IDENT	CLASS	DEFINITION	REFERENCES
LABVAL	VARIABLE	4978	4994 4995
LABWORD	VARIABLE	1219	1289 1290
LASTACT	VARIABLE	2399	2421 2427
LASTARG	VARIABLE	2825	2892 2909 2911 2912
LASTCOMP	VARIABLE	2491	2516 2518 2520 2524 2525
LASTRL	VARIABLE	3221	3248 3249 3250 3251 3251 3292 3294 3295 3295 3299 3301 3302 3303 3304
LBRACECH	CONSTANT	89	697 5271
LBRACKCH	CONSTANT	86	638 5266
LBRACKSY	CONSTANT	79	607 2509 3234 3240 3428 3433 3578 3631 3637 3665 3717 3848 3971 3987 4120 4192 4196
LEFT	VARIABLE	908	912 912 918 919 920 920 925 925 927 927
LEFT	VARIABLE	931	939
LEFTOPCO	VARIABLE	3775	3779 3798 3800 3802 3804 3806 3808 3809 3815 3815 3816 3832
LEFTOPCO	VARIABLE	3895	3897 3915 3916 3917 3918 3924 3924 3925 3930 3931 3932 3938 3938 3939 3955
LEFTOPTY	VARIABLE	3776	3813 3814 3816
LEFTOPTY	VARIABLE	3895	3922 3923 3925 3936 3937 3939
LEFTP	VARIABLE	5020	5060 5061 5090 5117 5125 5133 5142
LEFTPARA	VARIABLE	5020	5060 5068 5090 5117 5125 5133 5154 5154
LESSCHR	CONSTANT	88	644 5269
LETTER	CONSTANT	89	562 565 570 5260
LEVEL	VARIABLE	260	440 1058 1059 1090 1275 1313 1582 1618 1960 2003 2146 2154 2162 2666 2708 2740 2901 3023 3023 3031 3031 3062 3062 3067 3075 3075 3081 3101 3101 3130 3130 4204 4274 4330 4738 4738 4923 4940 4940 4952 4952 5064 5065 5151 5212 5214 5311 5435
LEVNR	FIELD	107	1058 1090 1275 1313 1551 1554 1581 1582 1617 1618 1960 2003 2146 2332 2423 2666 2708 2740 2901 3067 3081 3488 3492 4203 4204 4273 4274 4329 4330 4440 4440 5071 5139 5195 5203 5335 5341 5346 5351 5355 5360 5366 5374 5383 5391
LG	VARIABLE	1450	1452 1454 1455 1467
LINE	VARIABLE	203	440 442 469 5236
LINECNT	PROCEDUR	364	441 5455 5458 5467
LINK	VARIABLE	1243	1246 1252 1253 1259 1260
LINK	VARIABLE	1218	1273 1274 1275 1276 1280 1281 1284 1286 1287 1289 1290 1293 1295
LIT	PROCEDUR	945	1463 1512 1516 1570 1601 3114 3225 3268 3272 3448 3455 3456 3677 3693 3715 3798 3800 3802 3804 3806 3915 3916 3930 3978 3979 4056 4070 4072 4074 4076 4078 4080 4255 4310 4311 4341 4344 4345 4358 4359 4363 4433 4803 4805 4826 4833 4875 4879 4965 5061 5062 5081 5102 5145 5149 5156 5218 5220 5404 5405 5406 5407 5408 5409 5410 5411 5412 5413 5414 5415
LITBLANK	VARIABLE	312	1604 3495 4414 5409 5409
LITCOLON	VARIABLE	312	1592 3118 5408 5408
LITCOMMA	VARIABLE	312	1578 1585 3113 5080 5407 5407
LITERAL	VARIABLE	946	961
LITINTEG	VARIABLE	1450	1452 1453 1461 1468 1470
LITMINUS	VARIABLE	3220	3225 3225 3278 3378
LN	VARIABLE	1036	1038 1041 1044 1046
LNCNT	VARIABLE	209	361 365 365 366
LNPOS	VARIABLE	204	372 372 446 453 461 461 469 5235
LOC	PROCEDUR	852	900 918 937 956 979 1460 1483 1484 1533 1553 1569 1600 1636 3103 3103 3104 3189 3224 3224 3225 3243 3302 3396 3418 3447 3677 3693 3779 3779 3839 3897 3897 3962 4039 4111 4182 4182 4182 4182 4183 4183 4478 4802 4805 4813 4813 4814 4814 4862 4862 4863 4864 4864 4964 5059 5059 5059 5060 5060 5060 5063 5216 5216 5404 5405 5406 5407 5408 5409 5410 5411 5412 5413 5414 5415 5416
LOCSCNR	FIELD	171	2624 2625 5040
LOCTYP	FIELD	170	2621 5039 5042 5044 5045
LOWBND	FIELD	128	1060 1075 1116 1117 1356 1357 1366 1366 2056 2155 2163 3265 3281 3287 5111 5336 5347 5361 5368 5376 5377 5385 5386
LOWORD	CONSTANT	64	5259 5361
LPARACHR	CONSTANT	86	604 5265
LPARASYM	CONSTANT	79	635 2124 2128 2258 2598 2603 2979 2982 3393 3430 3478 3578 3631 3637 3665 3725 3848 3971 3987 4120 4192 4288 4295 4300 4391 5187
M	VARIABLE	336	337 337 340 341 341
MATCH	VARIABLE	413	
MATCHID	VARIABLE	2316	2336 2340 2342 2344 2346 2346 2350 2352

```
MATCHNR   VARIABLE   1002      1005 1016 1022 1026 1026
MATCHTYP  VARIABLE   1892      1911 1918 1919 1920 1923
MATCHTYP  VARIABLE   2237      2256
MATCHTYP  VARIABLE   2273      2291 2292
MATCHTYP  VARIABLE   2314      2353 2354 2356 2360
MATCHTYP  VARIABLE   2445      2473 2475 2476
MATCHTYP  VARIABLE   4547      4566
MATCHTYP  VARIABLE   4582      4600 4601
MATCHTYP  VARIABLE   4769      4853 4854 4856 4857
MAXCODLN  CONSTANT     56      735
MAXCOMPN  CONSTANT     60      288 1380
MAXINT    CONSTANT  STANDARD   5336 5336 5368 5368 5376 5376 5377 5377 5385 5385 5386 5386
MAXLG     FIELD       184      759 761 787 788 790 800 801 803 841 861 866 877 887 887 893 912 949 973 1458 1532
MAXLNL    CONSTANT     55      203 204 360 370 372 442 453 5236 5449
MAXNRERR  CONSTANT     59      387
MAXNRID   CONSTANT     52      96 421 5304
MAXPAGEL  CONSTANT     57      209 366
MAXSTRGL  CONSTANT     53      222 223 669 682 1076 1836 1839 1843 3688 3691
MAXWSL    CONSTANT     54      183 299 750 773 808 868 1480
MID       VARIABLE    483      486 487 488 489 490 493
MINUS     CONSTANT     85      498 499 595 5263
MODSYM    CONSTANT     74      583 589
MSIGN     VARIABLE   1766      1821 1827
MULTOPER  CONSTANT     82      590 599 3237 3366 3430 3436 3788 3793
MUSTBE    VARIABLE   1402      1414
N         VARIABLE    333      337 337
N         VARIABLE   1448      1452
NAMELST   VARIABLE   4179      4183 4255 4261 4262 4269 4270 4278 4279 4284 4311 4317 4318 4325 4326 4335 4336 4342 4343 4347 4347
                               4348 4349 4356 4357 4361 4362 4364 4364 4365 4366 4413 4414 4423 4425 4449 4450 4453 4454 4460 4471
NEST      VARIABLE    262      440 2529 2529 2560 2560 4500 4500 4607 4607 4657 4657 4798 4798 4846 4846 4892 4892 5053 5053 5311
NEW       VARIABLE   1144      1148 1149 1150 1150 1150 1181 1182 1183 1183 1183
NEW       VARIABLE   1223      1229 1230 1232 1233 1235 1235 1238
NEW       VARIABLE   2275      2301 2302 2303 2303 2303
NEW       VARIABLE   2444      2462 2463 2463 2463 2463
NEW       PROCEDUR  STANDARD   836 872 1055 1055 1061 1069 1072 1101 1193 1263 1309 1309 1310 1310 1311 1311 1314 1321 1332 1684
                               1788 1795 1802 1832 1850 1876 1886 1928 1958 1961 1966 2001 2004 2008 2040 2046 2051 2053 2060 2088
                               2094 2102 2142 2143 2147 2161 2183 2212 2331 2333 2421 2424 2521 2528 2568 2576 2614 2621 2639 2664
                               2667 2672 2706 2709 2713 2738 2741 2895 2901 2907 3064 3065 3068 3078 3079 3082 3200 3294 3298 3346
                               3503 4125 4402 4740 4942 4944 5193 5193 5195 5201 5201 5203 5327 5328 5333 5333 5339 5339 5344 5344
                               5350 5354 5358 5358 5364 5364 5364 5371 5371 5372 5372 5380 5380 5381 5381 5389 5389 5397
NEWACT    VARIABLE   1187      1190 1192 1192 1194 1195 1196
NEWACT    VARIABLE   2926      2949 2950 2953 2953 2953
NEWARG    VARIABLE   2825      2907 2907 2908 2908 2910 2911 2912
NEWARG    VARIABLE   2925      2949 2950 2952 2952 2952
NEWCONS   VARIABLE   4584      4611 4612 4613 4614 4615 4615
NEWLINK   VARIABLE   1218
NEWPAGE   PROCEDUR    354      366 5424 5439 5463
NEWX      VARIABLE   1084      1101 1101 1102 1102
NEWX      VARIABLE   1300      1332 1332 1333 1336 1337
NEWX      VARIABLE   3222      3346 3346 3347 3348
NEWX      VARIABLE   4775      4944 4944 4945 4945
NEXSTA    CONSTANT    476      549
NEXTCH    PROCEDUR    436      515 554 555 557 564 572 596 599 605 607 610 616 619 627 631 633 639 641 645 646 647 652 653 658 659
                               671 673 693 695 698 698 701 705 5434
NEXTF     FIELD       172      2626 5047
NILSYM    CONSTANT     74      3578 3631 3637 3665 3710 3848 3971 3987 4120
NILTYP    VARIABLE    282      3714 5328 5328 5328
NONIDCON  PROCEDUR   1764      1881 2168
```

IDENT	CLASS	DEFINITION	REFERENCES
NONIDCOR	PROCEDUR	1768	1823
NOTFND	VARIABLE	483	484 485 491 494
NOTFND	VARIABLE	1223	1231 1232 1234
NOTFND	VARIABLE	4584	4611 4612 4616 4617
NOTSKIP	VARIABLE	255	1004 1023 1027 1037 1086 1411 1670 1689 1721 1741 1760 1778 1802 1816 1850 1866 1886 1903 1934 1953 1965 1980 1993 2007 2022 2034 2074 2084 2110 2123 2182 2198 2232 2248 2269 2289 2296 2325 2360 2379 2386 2408 2418 2438 2459 2502 2514 2582 2597 2617 2639 2658 2671 2685 2698 2712 2725 2759 2768 2780 2793 2803 2815 2834 2881 2889 2918 2940 2992 3017 3024 3032 3035 3076 3135 3149 3165 3182 3208 3233 3375 3392 3406 3426 3532 3548 3563 3577 3586 3600 3616 3630 3647 3663 3672 3737 3787 3829 3847 3856 3905 3952 3970 3997 4047 4101 4119 4136 4191 4468 4491 4499 4526 4543 4558 4577 4598 4605 4649 4656 4678 4700 4715 4723 4755 4786 4906 4925 4972 4987 5011 5029 5051 5164 5190 5308
NOTSTAND	VARIABLE	293	623 2843 2854 2881 3036 5399
NOTSYM	CONSTANT	74	3578 3631 3637 3664 3668 3848 3971 3987 4120
NOTTHRU	VARIABLE	5018	5071 5072 5139 5140
NR	VARIABLE	1479	1485 1503 1503 1506
NR	VARIABLE	1631	1637
NRBUCK	CONSTANT	58	416 5306
NRFND	VARIABLE	1479	1494 1500 1500 1503 1503
NRKEYW	CONSTANT	63	213
NRLIT	VARIABLE	1481	1484 1506 1508 1509 1509 1513 1515
NXTACT	FIELD	113	1131 1192 1235 1236 1238 1240 1267 1280 1374 1375 1590 1626 2303 2305 2427 2463 2465 2478 2537 2544 2545 2904 2953 2959 3109 3334 4282 4339 4457 4745 4948 5137 5459
NXTARG	FIELD	160	1165 2908 2911 2952 2956 3506 3510 3523 4405 4409 4427 5366 5374 5383 5391
NXTEXP	FIELD	147	3204 3316 3509 3522 4383 4408 4426
NXTID	FIELD	155	1685 2137 2159 2428 2745 2913 5210
NXTONE	FIELD	164	1195 1281 1547
NXTREF	FIELD	152	1061 1102 1150 1151 1181 1183 1314 1332 1962 2005 2148 2334 2425 2668 2710 2741 2902 3069 3083 3347 4945 5196 5204
NXTSBND	FIELD	195	3250 3251 3294 3295 3304 4225 4234 4248
NXTSCID	VARIABLE	2030	2134 2136 2137 2137
NXTSTRG	FIELD	187	765 766 769 770 772 805 835 839 840 863 871 875 876
NXTVAL	FIELD	167	1930 4615 4619 4624 4625
OCC	FIELD	151	1061 1101 1154 1183 1317 1318 1336 1337 1962 2005 2147 2333 2424 2667 2709 2741 2902 3068 3082 3346 4944 5040 5196 5204
OFSYM	CONSTANT	75	2326 2338 2474 2513 2566 2574 3236 3365 3428 3434 3789 3819 3907 3942 4049 4092 4855
OLD	VARIABLE	1144	1148 1150 1150 1151 1153 1154 1181
OLD	VARIABLE	1223	1229 1235 1235 1236 1236
OLD	VARIABLE	2275	2301 2303 2304 2305
OLD	VARIABLE	2444	2462 2463 2464 2465
OLDACT	VARIABLE	2926	2949 2953 2958 2959
OLDARG	VARIABLE	2925	2949 2952 2955 2956
OLDARG	VARIABLE	3416	3506 3508 3510
OLDARG	VARIABLE	4180	4405 4407 4409
OLDCH	VARIABLE	477	618 622 631 632
OLDCONS	VARIABLE	4584	4611 4614 4618 4619 4623 4625
OLDER	VARIABLE	1144	1150 1151
OLDFLD	VARIABLE	1245	1263 1264 1265 1266 1267
OLDLEVEL	VARIABLE	4775	4923 4952 4952
OLDP	VARIABLE	750	792 796
OLDSEL	VARIABLE	3222	3330 3333 3344 3345 3347 3348
OMEGA	VARIABLE	315	5089 5124 5412 5412
OMEGAS	VARIABLE	5020	5059 5068 5089 5090 5116 5117 5124 5125 5132 5133 5155 5155
ORD	FUNCTION	STANDARD	415 415 415 416 514 514 517 517 1490 1491 1494 1496 1497 1500 1509 1510 1839 3691 5274 5443 5448 5450
ORDCNT	VARIABLE	2030	2134 2137 2137 2155 2163
ORDDBLDO	CONSTANT	68	533 5273
ORDEOLNC	CONSTANT	66	445 5237 5272
ORDEOSTR	CONSTANT	67	676 1184 5443 5448 5450

IDENT	CLASS	DEFINITION	REFERENCES
ORDER	VARIABLE	2030	2134 2140 2140 2146 2148
ORSYM	CONSTANT	75	3236 3365 3428 3434 3789 3819 3906 3927
OTHERCHR	CONSTANT	89	219 480 703 5259
OUT	PROCEDUR	728	1642 3128 5222
OUTPUT	VARIABLE	21	359 361 440 5443 5445 5445 5451 5452 5453 5455 5456 5458 5466 5466 5467 5469
PACKEDSY	CONSTANT	75	2599 2606
PAGENR	VARIABLE	208	356 356 360 5424
PARAMIDL	VARIABLE	2823	2841 2852 2865 2875 2893 2894 2897 2913 2913
PARAMTYP	VARIABLE	2823	2845 2856 2887 2890 2903 2907
PARM	FIELD	108	2536 2740 2899 2900 5084 5195 5203
PASSKND	FIELD	159	1163 2908 3508 3519 4407 4419 5366 5374 5383 5391
PERIODCH	CONSTANT	85	553 5262
PERIODSY	CONSTANT	80	559 3234 3320 3427 3433 4192 4196 5223
PI	VARIABLE	313	1586 1621 4277 4334 5410 5410
PIECE	VARIABLE	966	972 973 973 979 980 981 981 983 984 984 985 986 986 987 987
PIECE	VARIABLE	964	1483 1487 1488 1514 1516 1517 1518 1519
PLUS	CONSTANT	85	595 596 5263
PLUSMINU	CONSTANT	82	596 1817 1820 1867 1880 2124 2166 2249 2253 2598 2603 3237 3366 3430 3435 3579 3632 3637 3790 3820 3906 3911 3971 3975 4121 4559 4563
POINTER	CONSTANT	80	2600 2613 3234 3353 3430 3435 4192 4196
POP	PROCEDUR	1216	3130 4952 5214 5435
POS	VARIABLE	1480	1487 1488 1489 1490 1491 1492 1492 1494 1495 1495 1496 1497 1498 1500 1501 1501 1504 1504
PREFIX	VARIABLE	1634	1636 1637 1638 1638 1639 1641
PREFIX	VARIABLE	4476	4478 4503 4504 4504 4505 4512
PRINTTYP	PROCEDUR	1110	1122 1123 1156 1157 1158 1159 1164 1169
PRINTXRE	PROCEDUR	1142	5441
PRIORCOD	VARIABLE	3772	3808 3823
PRIORCOD	VARIABLE	3892	3917 3931 3946
PRIORCOD	VARIABLE	4033	4056 4082 4095
PRIORTYP	VARIABLE	3772	3812 3823
PRIORTYP	VARIABLE	3892	3921 3935 3946
PRIORTYP	VARIABLE	4033	4060 4086 4095
PROCFUND	PROCEDUR	3004	3155
PROCFUNP	PROCEDUR	3139	3157 5050
PROCID	CONSTANT	101	118 1095 1160 1311 1311 1323 1329 1545 1575 1609 3065 3079 3091 3460 4200 4265 4321 4387 4388 4395 4437 5075
PROCSYM	CONSTANT	75	1742 1754 1954 1974 1994 2016 2659 2679 2699 2719 2760 2774 2794 2809 2835 2869 3018 3021 3150 3153 5030 5310
PROGCOD	VARIABLE	5171	5216 5218 5219 5221 5222
PROGID	VARIABLE	211	357 357 5183 5185 5424
PROGNAME	VARIABLE	355	357 358 359
PROGRAMM	PROCEDUR	5168	5430
PROGSTMN	VARIABLE	319	5151 5217
PROGSYM	CONSTANT	75	5180
PSNAME	VARIABLE	1032	1040 1043
PTR	VARIABLE	1218	1276 1281
PTRACT	VARIABLE	1084	1093 1095 1096 1102 1102
PTRACT	VARIABLE	1111	1128 1129 1130 1131 1131
PTRACT	VARIABLE	1522	1529 1551 1554
PTRACT	VARIABLE	1766	1783 1784 1785 1786
PTRACT	VARIABLE	1857	1871 1872 1873 1874
PTRACT	VARIABLE	1941	1958 1959 1969 1969 1970 1970 2001 2002 2011 2011 2012 2012
PTRACT	VARIABLE	2028	2081 2086 2090 2091 2099 2100 2104 2142 2143 2144 2156
PTRACT	VARIABLE	2316	2344 2345 2347 2353
PTRACT	VARIABLE	2588	2628 2629 2630
PTRACT	VARIABLE	2646	2664 2665 2673 2674 2674 2675 2675 2706 2707 2714 2715
PTRACT	VARIABLE	2734	2738 2739 2742
PTRACT	VARIABLE	2824	2883 2884 2887 2895 2896 2904 2904

IDENT	CLASS	DEFINITION	REFERENCES
PTRACT	VARIABLE	3006	3064 3065 3066 3073 3078 3079 3080 3087 3090 3091 3092 3097 3098 3106 3108 3109 3110
PTRACT	VARIABLE	3415	3438 3439 3442 3446 3454 3457 3463 3464 3469 3472 3480 3481 3484 3485 3488 3488 3491 3491 3492 3492 3494 3511 3512 3512
PTRACT	VARIABLE	4176	4198 4199 4202 4203 4204 4205 4224 4259 4309 4315 4355 4387 4388 4390 4395 4396 4410 4411 4411 4413 4440
PTRACT	VARIABLE	4706	4724 4725 4726 4740 4741 4742 4743 4744
PTRACT	VARIABLE	4773	4907 4908 4911 4915 4926 4927 4928 4942 4943 4945 4946 4947 4959
PTRACT	VARIABLE	4978	4996 4998
PTRACT	VARIABLE	5017	5039 5040 5041 5042
PTRACT	VARIABLE	5170	5193 5193 5194 5199 5201 5201 5202 5207
PTRACT	VARIABLE	5233	5333 5333 5334 5337 5337 5339 5339 5340 5342 5344 5344 5345 5348 5348 5350 5351 5352 5354 5355 5356 5358 5358 5359 5362 5362 5364 5364 5364 5365 5369 5371 5371 5372 5372 5373 5378 5380 5380 5381 5381 5382 5387 5389 5389 5390 5393
PTRARG	VARIABLE	1144	1161 1162 1163 1164 1165 1165
PTRIDNR	FIELD	136	1133 2616 5039 5045 5198 5206 5328
PTRTOREC	VARIABLE	1245	1253 1254 1255 1255 1256 1259 1260 1261
PTRTYP	CONSTANT	102	136 1133 1379 1394 2614 3357 4379 5120 5193 5201 5328
PUSH	PROCEDUR	1187	1329 1716 1751 1970 2012 2156 2675 2715 2742 3073 3087 3109 3512 4411 4744 4947 5199 5207 5337 5342 5348 5352 5356 5362 5369 5378 5387 5393
PUSHVARS	PROCEDUR	2733	2770 2805
PUT	PROCEDUR	STANDARD	5445 5453 5466
QUOTE	CONSTANT	88	663 5270
RBRACKCH	CONSTANT	87	638 5266
RBRACKSY	CONSTANT	79	556 2075 2097 2199 2208 2512 3183 3196 3235 3317 3364 3428 3434 3549 3557 3601 3610 3631 3635 3723 3789 3818 3907 3941 4049 4091
READ	PROCEDUR	STANDARD	444 457 462
REC	VARIABLE	2314	2331 2332 2332 2333 2333 2334 2340 2355 2356 2360
RECEL1	VARIABLE	2492	2533 2534 2536 2537 2539 2541 2545 2545
RECEL2	VARIABLE	2492	2537 2538 2539 2544 2544
RECLST	VARIABLE	2237	2259 2264 2269
RECLST	VARIABLE	2364	2381 2387
RECLST	VARIABLE	2397	2419 2421 2421 2422 2427 2433 2438
RECORDSE	PROCEDUR	2397	2460
RECORDSY	CONSTANT	76	2503 2527 2599 2611 5310
RECTYP	CONSTANT	102	132 1125 1256 1370 1394 2528 3047 3325 4732 4934 5119
REFERTYP	FIELD	137	1382 1382 2617 2630 2631 3361 5042 5044 5197 5205 5328
REFID	VARIABLE	2026	2040 2046 2051 2061 2089 2095 2103 2149 2151 2154 2162 2183
REFID	VARIABLE	2489	2529
REFID	VARIABLE	1649	2605 2609 2612 2640
REFP	CONSTANT	111	2899
RELOPER	CONSTANT	82	97 215 585 645 652 3237 3366 3430 3436 3790 3820 3908 3943 4048 4065
REMWS	VARIABLE	303	780 784 786 786 790 790 793 796 797 797 803 803 808 841 842 843 843 868 877 878 879 879 5402
REPEATSY	CONSTANT	76	4788 4891 4989 5004 5310
REPLACE	VARIABLE	311	4236 5405 5405
REPLLST	VARIABLE	3214	3248 3298 3299 3318 3351
REPLLST	VARIABLE	4178	4210 4211 4218 4223
REPLMENT	VARIABLE	4179	4182 4219 4221 4241 4250 4261 4310 4317 4345 4346 4359 4360 4433 4443 4444 4448 4449 4451 4470
REPSTMLI	PROCEDUR	4634	4652 4895
RESET	PROCEDUR	STANDARD	1184 5464
RESULT	VARIABLE	1481	1483 1512 1513 1514 1515 1517 1519
RETACT	VARIABLE	1054	1055 1055 1056 1063
RETACT	VARIABLE	1300	1309 1309 1310 1310 1311 1311 1312 1329 1329 1331 1332 1333 1338
RETACT	VARIABLE	2819	2892 2904 2904 2916
RETARG	VARIABLE	2819	2892 2910 2916
RETCOD	VARIABLE	1448	1458 1459 1460 1463 1464 1466 1469 1471
RETCOD	VARIABLE	1522	1532 1533 1533 1537 1539 1553 1555
RETCOD	VARIABLE	1559	1570 1578 1579 1585 1586 1587 1592
RETCOD	VARIABLE	3773	3816 3823

```
RETCOD     VARIABLE   3893       3925 3939 3946
RETCOD     VARIABLE   4034       4056 4056 4057 4063 4063 4070 4072 4074 4076 4078 4080 4082 4083 4089 4089 4095
RETNR      VARIABLE   1002       1012 1023 1024 1027 1028
RETRIEVE   VARIABLE    311       3307 4227 5404 5404
RETTYP     VARIABLE   1068       1069 1070 1079
RETTYP     VARIABLE   2026       2040 2040 2046 2046 2051 2051 2053 2054 2055 2056 2056 2060 2061 2088 2089 2094 2095 2102 2103 2104
                                 2161 2162 2162 2163 2163 2183 2183
RETTYP     VARIABLE   2489       2516 2521 2522 2523 2524 2528 2528 2529 2531 2567 2568 2568 2575 2576 2576 2582
RETTYP     VARIABLE   1649       2605 2609 2612 2614 2616 2617 2621 2630 2631 2639 2640
RETTYP     VARIABLE   3773       3816 3823 3829 3831
RETTYP     VARIABLE   3893       3925 3939 3946 3952 3954
RETTYP     VARIABLE   4034       4061 4062 4087 4088 4095 4101 4103
RETTYP     FIELD       119       1169 1325 3071 3072 3085 3086 3472 3485 3511 4205 4396 4410 5367 5372 5375 5375 5376 5376 5381 5384
                                 5384 5385 5385 5392
RETTYPAC   VARIABLE   3006       3041 3042 3043
RETTYPID   VARIABLE   3007       3037 3038 3039 3041 3045
RETURNW    VARIABLE    392       393 394
REWRITE    PROCEDUR  STANDARD    1145 5425 5426
RIGHT      VARIABLE    908       912 918 922 924 926 927
RIGHT      VARIABLE    931       936 937 937 941
RIGHTP     VARIABLE   5020       5060 5062 5090 5117 5125 5133 5142
RPARACHR   CONSTANT     86       638 639 5265
RPARASYM   CONSTANT     79       639 1671 1680 2075 2097 2164 2250 2260 2262 2284 2294 2374 2383 2409 2431 2454 2457 2935 2945 2988
                                 3183 3196 3234 3364 3402 3427 3433 3731 3788 3818 3906 3941 4048 4091 5211
RSTRG      VARIABLE    746       835 836 838 839 840 841 841 842
RSTRG      VARIABLE    852       859 860 862 865 866 867 867 871 872 874 875 876 877 877 878 881
RUNACT     VARIABLE   3416       3501 3502 3507 3509 3509
RUNACT     VARIABLE   4180       4400 4401 4406 4408 4408
RUNARG     VARIABLE   3416       3503 3505 3506 3507 3508 3508
RUNARG     VARIABLE   4180       4402 4404 4405 4406 4407 4407
SAVE       VARIABLE    750       769 770 770
SCALID     CONSTANT    101       115 1055 1057 1156 1286 1309 1309 1320 1575 1609 1783 1784 1871 1872 1958 1961 2001 2004 2081 2086
                                 2142 3440 4200 4265 4321 4437 4959 4996 5075 5350 5354
SCALTYP    CONSTANT    102       126 1055 1057 1072 1114 1352 1393 1416 1416 1432 1432 1917 1918 2035 2036 2053 2143 2161 2213 2347
                                 3263 3443 4849 4911 5087 5109 5333 5344 5358 5364 5372 5372 5381 5381
SCANACTL   VARIABLE   1526       1541 1543 1545 1547 1547
SCIDLST    VARIABLE   2028       2132 2134 2139 2141 2145 2159 2159
SCIDNR     FIELD       126       1059 1073 1115 1352 1352 1354 1354 1364 1364 1418 1418 1434 1434 1919 1919 2042 2042 2054 2054 2154
                                 2162 5336 5347 5361 5368 5375 5377 5384 5386
SCLEVNR    FIELD       127       1059 1074 1353 1353 1365 1365 1419 1419 1435 1435 1920 1920 2043 2043 2055 2055 2154 2162 5336 5347
                                 5361 5368 5375 5377 5384 5386
SCNR       VARIABLE    207       379 440 641 641 1061 1101 1317 1318 1336 1337 1962 2005 2147 2333 2424 2624 2625 2667 2709 2769
                                 2804 2902 3022 3030 3346 4944 5196 5204 5235
SEARCHAL   FUNCTION   1299       1338 1783 1871 2081 2099 2344 2628 2883 3041 3438 3480 4198 4309 4355 4387 4724 4907 4926 4959 4996
                                 5039
SEARCHON   FUNCTION   1082       1103 1105 1107 1716 1751 1970 2012 2141 2675 2715 2737 2894 3063 3076 5199 5207
SECLAST    VARIABLE   4178       4223 4234 4247 4248 4251
SECLST     VARIABLE   3222       3329 3331 3332 3333 3334 3334
SECLST     VARIABLE   4706       4737 4739 4741 4743 4745 4745
SECLST     VARIABLE   4775       4939 4941 4943 4946 4948 4948
SECLST1    VARIABLE   1344       1371 1372 1373 1374 1374 1377
SECLST2    VARIABLE   1344       1371 1372 1373 1375 1375 1377
SECOPCOD   VARIABLE   4036       4039 4059 4063 4085 4089 4104
SECOPTYP   VARIABLE   4036       4059 4060 4085 4086
SECTNS     FIELD       132       1128 1261 1371 1371 2531 3329 4737 4939
SELTYP     VARIABLE   4769       4848 4849 4850 4854
SEMICCHR   CONSTANT     87       641 5267
SEMICSYM   CONSTANT     79       641 1316 1335 1671 1680 1708 1719 1971 2013 2075 2097 2250 2262 2284 2287 2374 2377 2409 2431 2454
```

IDENT	CLASS	DEFINITION	REFERENCES
			2457 2623 2676 2716 2771 2806 2935 2938 2979 2990 3026 3057 3156 3234 3364 3427 3433 3788 3818 3906 3941 4048 4091 4193 4288 4486 4489 4527 4536 4560 4570 4593 4596 4644 4647 4788 4961 4988 5003 5213 5309
SETELEME	PROCEDUR	3567	3607 3640
SETELEML	PROCEDUR	3590	3608 3641
SETELEMR	PROCEDUR	3538	3582
SETRANGE	PROCEDUR	3620	3722
SETSYM	CONSTANT	76	2503 2570 2600 2611
SEXPCOD	VARIABLE	3958	3978 3979 3985 3985 3991
SEXPTYP	VARIABLE	3958	3983 3984 3991
SH	VARIABLE	3220	3262 3270 3271 3274 3275 3278 3278 3279 3283 3284 3301 3307 3307 3308 3309 3310 3311 3311
SHIFTNR	VARIABLE	3220	3224 3268 3270 3272 3274 3282 3283 3378
SHOULDBE	VARIABLE	3776	3797 3799 3801 3803 3805 3812
SHOULDBE	VARIABLE	4036	4069 4071 4073 4075 4077 4079 4086
SIGNAL	VARIABLE	3222	3330 3331 3332 3335
SIMPEXPR	PROCEDUR	4033	4130
SIMPLEEX	PROCEDUR	3958	4059 4085 4129
SIMPLETY	PROCEDUR	2026	2205 2510 2575 2605
SIMPSTMR	PROCEDUR	4173	4795
SIMPTYPL	PROCEDUR	2187	2206 2511
SIMPTYPR	PROCEDUR	2064	2176
SIZE	VARIABLE	746	773 808 841 843
SIZE	VARIABLE	852	857 857 861 866 868 877 879 881
SIZE	VARIABLE	945	949 956 958 959 960
SLASH	CONSTANT	86	598 5264
SLICES	VARIABLE	4179	4182 4224 4232 4233 4243 4244 4250 4470
SORT	PROCEDUR	1221	1246
SORTALL	PROCEDUR	1243	1267 1284
SORTIN	VARIABLE	1221	1233 1236 1237 1238 1240 1240
SOURCE	VARIABLE	890	893 900 902 903 904 905
SPBOOLEA	VARIABLE	275	5315 5346 5347
SPCHAR	VARIABLE	275	5315 5360 5361
SPCHR	VARIABLE	276	3491 5320 5391
SPCONST	VARIABLE	278	1789 1803 1851 1877 1887 1967 2009 5322
SPELLED	VARIABLE	1036	1038 1046
SPFALSE	VARIABLE	275	5316 5355
SPFILE	VARIABLE	278	2568 5318
SPFORWAR	VARIABLE	278	3027 3058 5322
SPGET	VARIABLE	276	4299 4307 5320
SPINPUT	VARIABLE	275	4309 5121 5192 5195 5199 5317
SPINTEGE	VARIABLE	275	1073 1418 1418 5198 5206 5314 5335 5336
SPLABEL	VARIABLE	278	1059 5317
SPNEW	VARIABLE	276	4372 5321
SPORD	VARIABLE	276	3488 5319 5366
SPOUTPUT	VARIABLE	276	4355 5122 5200 5203 5207 5318
SPPRED	VARIABLE	276	3491 5323 5374
SPPUT	VARIABLE	276	4299 4353 5321
SPREAD	VARIABLE	277	4290 5325
SPREADLN	VARIABLE	277	4291 5325
SPREAL	VARIABLE	275	1796 1833 3709 5314 5341 5341
SPSET	VARIABLE	278	2576 5319
SPSUCC	VARIABLE	276	3492 5323 5383
SPTRUE	VARIABLE	275	5316 5351
SPUNCH	VARIABLE	305	738 741 743 5184 5186 5425
SPWRITE	VARIABLE	277	4290 5324
SPWRITEL	VARIABLE	277	4291 5324
SRCHKW	FUNCTION	482	493 494 580
SRCHTAB	VARIABLE	258	1089 1090 1093 1195 1196 1276 1281 1302 1331 1541 3090 3709 5305

IDENT	CLASS	DEFINITION	REFERENCES
SSS	PROCEDUR	890	919 937 939 980 3303 3369 3823 3946 4095 4219 4221 4227 4236 4250
ST	VARIABLE	497	498 498 499 500
STACK	VARIABLE	264	1192 1192 1273 1280 1295 1571 1605 4256 4312 4434 5069 5311
STACKTRA	VARIABLE	1567	1571 1572 1574 1577 1581 1582 1584 1590 1590
STACKTRA	VARIABLE	1598	1605 1606 1608 1612 1617 1618 1620 1626 1626
STACKTRA	VARIABLE	4177	4256 4257 4259 4264 4268 4273 4274 4276 4282 4282 4312 4313 4315 4320 4324 4329 4330 4333 4339 4339 4434 4435 4436 4440 4442 4452 4457 4457
STACKTRA	VARIABLE	4767	
STACKTRA	VARIABLE	5017	5069 5071 5074 5078 5084 5086 5096 5106 5121 5122 5130 5137 5137 5138 5139
STAR	CONSTANT	86	598 599 5264
STATE	VARIABLE	477	514 516 549 549 550
STATEMEN	PROCEDUR	1655	4494 4534 4568 4651 4687 4694 4801 4822 4872 4894 4951 4976 5152
STEND	VARIABLE	1480	1488 1489 1498
STM	VARIABLE	314	1638 3120 4504 4829 4836 4878 4881 5411 5411
STMCOD	VARIABLE	1631	1640 1641 1641 1642
STMCOD	VARIABLE	3008	3104 3106 3125 3126 3127 3128 3128
STMCOD	VARIABLE	4179	4183 4254 4284 4285 4286 4306 4349 4350 4351 4366 4367 4368 4459 4460 4461 4462 4470
STMCOD	VARIABLE	4764	4802 4803 4804 4806 4806 4807 4808 4808 4814 4825 4841 4842 4843 4843 4864 4874 4887 4888 4889 4889 4964 4965 4966 4966
STMNR	VARIABLE	261	440 3119 4286 4351 4368 4462 4493 4533 4800 4821 4871 4966 4991 4991 5053 5053 5150 5311
STMOUT	PROCEDUR	1631	4286 4351 4368 4462 4808 4843 4889 4966 5159
STMREST	VARIABLE	3008	3103 3105 3112 3113 3114 3115 3117 3118 3121 3122 3123 3126 3126
STMTAIL	VARIABLE	4766	4814 4826 4827 4828 4830 4831 4838 4839 4840 4841 4841 4864 4875 4876 4877 4884 4885 4886 4887 4887
STM1NR	VARIABLE	4766	4821 4829 4843
STM2NR	VARIABLE	4766	4823 4832 4835
STRG	VARIABLE	728	733 736 738
STRGDESC	TYPE	182	187 144 189 301 302 750 854 910 933 969
STRGEND	VARIABLE	479	668 677 688
STRGLG	VARIABLE	478	668 689 689 689
STRGPTR	VARIABLE	223	668 669 676 679 681 682 687 687 689 5302
STRING	VARIABLE	222	676 679 681 1839 3691
STRINGSY	CONSTANT	81	668 1817 1835 1867 1880 2124 2166 2249 2253 2599 2604 3578 3632 3637 3664 3686 3848 3972 3987 4120 4559 4563
STYP	FIELD	115	1055 1056 1156 1320 1785 1873 1969 2011 2090 2143 2144 3442 5352 5356
SUB	FIELD	193	3301 4228 4237 4246
SUBRANGE	PROCEDUR	2032	2092 2171
SUBSBNDS	TYPE	192	195 307 3214 3221 4178
SUBSCRIP	VARIABLE	1527	1553 1554 1555 1555
SUPERSCR	VARIABLE	320	624 972 5418
SW	VARIABLE	1894	1919 1921 1922
SWASSIGN	VARIABLE	3213	3246 3290 3318 3351
SWEOLN	VARIABLE	205	446 450 470 5235
SWERR	VARIABLE	1343	1346 1350 1355 1358 1363 1368 1368 1376 1389
SWERR	VARIABLE	1409	1412 1417 1420 1421 1425 1433 1436 1437 1441 1441
SWERRMSG	VARIABLE	234	377 5433 5434
SWFND	VARIABLE	1084	1088 1090 1091
SWFORW	FIELD	120	1096 1170 1326 3069 3083 3092 3097 5367 5373 5382 5392
SWPACK	VARIABLE	2187	2206 2212
SWPACK	VARIABLE	2489	2511 2522 2528
SWPACKA	FIELD	131	1071 1120 1368 1368 2212 2522
SWPACKR	FIELD	133	1126 1376 1376 2528
SWPLUS	VARIABLE	336	337 338
SWTRUNC	VARIABLE	479	569 572 573
SWVAR	FIELD	178	3192 3399 3441 3467 3479 3669 3687 3702 3708 3712 3718 3730 3730 3795 3913 3929 3976 4055 4067 4128
SWVAR	FIELD	146	3203 3518 4380 4418
SWXRFPRR	FIELD	134	1259 1260 2529
SYNCHRON	PROCEDUR	995	1021 1689 1760 1802 1850 1886 1934 1980 2022 2110 2182 2232 2269 2360 2438 2582 2639 2685 2725 2780 2815 2918 3135 3165 3208 3375 3406 3532 3563 3586 3616 3647 3737 3829 3856 3952 3997 4101 4136 4468

IDENT	CLASS	DEFINITION	REFERENCES
			4543 4577 4700 4755 4972 5011 5164
SYNCHRSY	VARIABLE	253	996 1026 5309
TAGFIELD	PROCEDUR	2314	2473
TAIL	VARIABLE	750	757 763 765 766 772 772 837 839
TAIL	VARIABLE	854	859 863 873 875
TARGET	VARIABLE	890	893 899 900 902 905
TARGET	VARIABLE	931	936 939 941 942
TARGET	VARIABLE	945	949 955 956 958 961
TERM	PROCEDUR	3835	3920 3934 3981 3990
TERMCOD	VARIABLE	3835	3852
TERMIN	VARIABLE	317	1640 3127 5414 5414
TERMINAL	PROCEDUR	1002	1677 1714 1749 1964 1971 2000 2006 2013 2131 2164 2169 2257 2258 2260 2336 2416 2472 2474 2509 2512 2513 2560 2566 2574 2616 2670 2676 2705 2711 2716 2767 2771 2800 2802 2806 2842 2846 2851 2853 2857 2864 2866 2867 2874 2988 3023 3026 3031 3034 3038 3057 3156 3317 3323 3402 3723 3731 4213 4567 4686 4693 4722 4820 4855 4870 4905 4917 4924 4957 5000 5052 5183 5187 5188 5211 5213
TERMLIST	PROCEDUR	3892	3925 3939 3991
TERMTYP	VARIABLE	3835	3852 3856
TEXT	VARIABLE	369	381 382 383 385
TEXT	VARIABLE	397	400
THENSYM	CONSTANT	76	3236 3365 3429 3435 3790 3819 3908 3942 4050 4092 4820
TNR	VARIABLE	392	393 393 394
TOKENDES	PROCEDUR	392	401 1016 1018
TOKENFD	VARIABLE	239	508 514 517 517 582 585 587 588 589 596 599 600 646 647 648 653 654 669 689 689 1012 1024 1027 1783 1787 1793 1796 1821 1830 1833 1836 1839 1842 1843 1871 1875 1960 2174 2413 2666 2764 2839 3027 3058 3682 3688 3691 3697 3704 3796 3914 3977 4068 4793 4994
TOKENNR	VARIABLE	238	508 518 519 527 532 535 555 556 559 580 581 583 584 585 586 590 596 599 607 635 639 641 645 652 659 660 668 701 709 996 997 1005 1018 1022 1026 1316 1335 1671 1673 1688 1708 1710 1727 1742 1745 1759 1779 1781 1801 1817 1819 1849 1867 1869 1885 1904 1906 1933 1954 1956 1979 1994 1996 2021 2075 2078 2109 2124 2127 2181 2199 2201 2231 2249 2252 2268 2284 2286 2309 2326 2328 2359 2374 2376 2392 2409 2411 2437 2454 2456 2484 2503 2505 2581 2598 2602 2623 2638 2659 2661 2684 2699 2701 2724 2760 2762 2779 2794 2796 2814 2835 2837 2917 2935 2937 2963 2979 2981 2999 3018 3020 3027 3058 3134 3150 3152 3164 3183 3185 3207 3234 3239 3374 3393 3405 3427 3432 3531 3549 3551 3562 3578 3585 3601 3603 3615 3631 3634 3646 3664 3667 3736 3788 3792 3828 3848 3855 3906 3910 3951 3971 3974 3996 4048 4052 4100 4120 4135 4192 4195 4295 4300 4391 4467 4486 4488 4509 4527 4529 4542 4559 4562 4576 4593 4595 4629 4644 4646 4662 4679 4681 4699 4716 4718 4754 4787 4791 4971 4988 4992 5010 5030 5163 5180 5223 5227
TOKENRG	TYPE	97	238 253 254 392 397 1002
TOKW	VARIABLE	215	394 5251 5252 5252 5252 5253 5253 5253 5254 5254 5254 5255 5255 5255 5256 5256 5256 5257 5257 5257
TOP	VARIABLE	483	484 485 486 490
TOPBOOL	VARIABLE	289	3192 3399 3441 3467 3479 3669 3687 3702 3708 3712 3718 3730 3730 3795 3913 3929 3976 4055 4067 4124 4125 4126 4126 4127 4128 4128 4128 4132 4132 5397 5397 5397
TOSYM	CONSTANT	76	3236 3365 3429 3434 3789 3819 3907 3942 4049 4092 4679 4682
TRANSIT	VARIABLE	221	516 549 5276 5276 5276 5277 5277 5278 5278 5278 5279 5279 5280 5280 5280 5281 5281 5282 5282 5282 5283 5283 5284 5284 5284 5285 5285 5286 5286 5286 5287 5287 5288 5288 5288 5289 5289 5290 5290 5290 5291 5291 5292 5292 5292 5293 5293 5294 5294 5294 5295 5295 5296 5296 5296 5297 5297 5298 5298 5298 5299 5299
TRAV	VARIABLE	750	757 758 759 761 764 765 766 766 766 769 769 769 780 781 784 786 787 787 788 788 790 792 793 794 795 800 800 801 801 803 803 805 805
TRAV	VARIABLE	854	859 860 861 862 863 863 863
TRAVRL	VARIABLE	4178	4223 4225 4228 4230 4234 4234 4234 4237 4239 4246 4246
TRUE	CONSTANT	STANDARD	484 572 677 1027 1071 1105 1231 1255 1350 1355 1358 1417 1420 1425 1433 1436 1721 1921 2289 2296 2379 2386 2459 2609 2849 2940 2992 3030 3069 3193 3245 3362 3400 4128 4211 4491 4499 4598 4605 4611 4649 4656 4730 4932 5035 5308 5399 5399 5434
TUPINIT	VARIABLE	5020	5059 5102 5104 5105 5112 5113 5116 5142
TYP	FIELD	117	1157 1158 1159 1253 1266 1266 1321 1321 1322 1373 1373 2104 2347 2353 2356 2425 2630 2674 2714 2742 2887 2903 3043 3345 3463 3709 4202 4726 4743 4743 4911 4915 4928 4946 4946 5042 5086 5096 5106 5193 5194 5201 5202 5333 5334 5337 5339 5340 5344 5345 5348 5358 5359 5362
TYPBOOLE	VARIABLE	281	1423 1423 3266 3453 3673 3676 4061 4087 4817 4867 4898 5348 5352 5356
TYPCHAR	VARIABLE	281	1077 1838 3444 3690 5362 5392

IDENT	CLASS	DEFINITION	REFERENCES
TYPCLASS	TYPE	102	125
TYPEDEFP	PROCEDUR	2644	5035
TYPEDFPT	PROCEDUR	2648	2677 2717
TYPEPROC	PROCEDUR	1649	2419 2515 2567 2586 2674 2714 2770 2805 2845 2856 3036
TYPES	TYPE	124	129 130 137 119 117 115 145 158 170 272 281 282 283 1066 1068 1110 1245 1341 1402 1649 1653 1764 1855 1891 1892 1894 1941 2026 2029 2187 2187 2189 2189 2237 2239 2273 2314 2399 2445 2489 2491 2646 2731 2823 3006 3169 3172 3216 3217 3219 3383 3411 3413 3651 3772 3773 3776 3835 3837 3892 3893 3895 3958 3960 4033 4034 4036 4109 4176 4547 4549 4582 4667 4669 4706 4766 4769 4769 4773 4775 5017
TYPESYM	CONSTANT	77	1742 1754 1954 1974 1994 2016 2699 2702 5030
TYPID	CONSTANT	101	117 1157 1252 1259 1260 1309 1309 1321 1575 1609 2099 2100 2344 2345 2628 2629 2664 2706 2883 2884 3041 3042 3460 4200 4265 4321 4437 5039 5041 5075 5333 5339 5344 5358
TYPIN	VARIABLE	3216	3254 3255 3258 3260 3263 3265 3266 3281 3286 3287 3314 3314 3318 3324 3325 3328 3329 3342 3345 3349 3351 3356 3357 3360 3361 3361 3362 3369
TYPIN	VARIABLE	3413	3442 3443 3444 3453 3461 3463 3472 3476
TYPIN	VARIABLE	4176	4201 4202 4205 4208 4212
TYPIN	VARIABLE	4706	4726 4728 4731
TYPIN	VARIABLE	4775	4928 4930 4933
TYPINTEG	VARIABLE	281	1792 1829 2222 3703 5197 5205 5337 5367 5393
TYPKLASS	FIELD	125	1057 1113 1254 1256 1347 1347 1349 1349 1351 1361 1362 1392 1413 1413 1416 1416 1432 1432 1824 1917 1918 1923 1924 2035 2036 2037 2038 2213 2214 2347 3047 3191 3254 3255 3263 3324 3325 3356 3357 3398 3443 3672 3729 4378 4732 4733 4816 4849 4850 4866 4897 4911 4934 4935 5086 5097 5107 5109
TYPOUT	VARIABLE	3217	3318 3351 3362 3369 3375 3377
TYPOUT	VARIABLE	3411	3476 3483 3485 3499 3513 3532 3534
TYPOUT	VARIABLE	4176	4212 4215
TYPOUT	VARIABLE	4706	4731 4732 4733 4737
TYPOUT	VARIABLE	4775	4933 4934 4935 4939
TYPTYP	VARIABLE	2646	2672 2673 2674 2674 2713 2713 2714 2714
TYP1	VARIABLE	1341	1347 1349 1351 1352 1353 1354 1356 1357 1360 1361 1364 1365 1366 1367 1368 1371 1376 1379 1382 1392 1397
TYP1	VARIABLE	1402	1413 1416 1418 1419 1423 1423 1428 1432 1434 1435
TYP2	VARIABLE	1341	1347 1349 1352 1353 1354 1356 1357 1360 1362 1364 1365 1366 1367 1368 1371 1376 1379 1382 1397
TYP2	VARIABLE	1402	1413 1416 1418 1419 1428 1432 1434 1435
UNDEF	CONSTANT	102	138 1134 1321 1347 1347 1361 1362 1413 1413 1788 1795 1802 1832 1850 1876 1886 1923 1924 1966 2008 2037 2038 2040 2046 2051 2060 2088 2094 2102 2183 2214 2347 2568 2576 2639 2672 2713 3191 3254 3324 3356 3398 3672 3729 4379 4733 4816 4850 4866 4897 4935 5087 5327 5339
UNDFIDNR	FIELD	138	1135 1136 1321 1789 1796 1803 1833 1851 1877 1887 1967 2009 2040 2046 2051 2061 2089 2095 2103 2183 2568 2576 2640 2673 2713 5327 5341
UNDFZERO	VARIABLE	283	1320 1322 2218 2225 2354 2360 2418 2514 2523 2582 2617 2631 2768 2803 2890 3035 3040 3044 3055 3072 3086 3258 3328 3342 3349 3360 3375 3377 3461 3483 3511 3513 3532 3534 3722 3737 3814 3829 3831 3856 3923 3937 3952 3954 3984 4062 4088 4101 4103 4136 4138 4201 4208 4410 4728 4853 4910 4914 4916 4930 5044 5327 5327 5328
UNLABLST	PROCEDUR	4759	5001 5006
UNPKSTRT	PROCEDUR	2489	2609 2612
UNSGINTE	CONSTANT	81	518 527 532 1714 1749 1779 1792 1817 1829 1867 1880 2124 2166 2249 2253 2598 2603 3579 3632 3638 3664 3700 3849 3972 3988 4121 4559 4563 4957 4988 4993
UNSGREAL	CONSTANT	81	519 535 1779 1795 1817 1832 1867 1880 2124 2166 2249 2253 2598 2603 3579 3632 3638 3664 3707 3849 3972 3988 4121 4559 4563
UNTILSYM	CONSTANT	77	3236 3366 3429 3435 3790 3819 3908 3942 4050 4092 4193 4288 4527 4536 4644 4654 4789 4961 4990 5005 5309
UNUSED	VARIABLE	884	887 887
UP	FIELD	179	3192 3399 3730 4124 4125 4126 4127 4127 4128 5397
USED	VARIABLE	751	757 761 761 773
VALEE	FIELD	167	1929 4613 4613
VALP	CONSTANT	110	2900
VALPREF	VARIABLE	318	3111 5131 5415 5415
VALUS	TYPE	167	167 1892 1894 2239 4547 4581 4581 4584 4584 4769
VALUU	FIELD	116	1060 1290 1320 1786 1874 1969 2011 2091 2146 2148 3446 3454 3457 4998 5352 5356
VARB	CONSTANT	109	2536 2740 5084 5195 5203

IDENT	CLASS	DEFINITION	REFERENCES
VARDCLPA	PROCEDUR	2729	5049
VARDCLPT	PROCEDUR	2749	2772 2807
VARIANT	PROCEDUR	2237	2291 2475
VARIANTL	PROCEDUR	2273	2292 2476
VARID	CONSTANT	101	117 1158 1252 1310 1310 1322 1545 1576 1610 2738 2895 3438 3462 4198 4202 4266 4309 4322 4355 4438 4724 4725 4907 4908 4926 4927 5076 5193 5201
VARIDLST	VARIABLE	2731	2736 2737 2740 2745 2745 2766 2801
VARSELEC	PROCEDUR	3213	3318 3351 3362 3475 4211 4730 4932
VARSW	VARIABLE	2824	2839 2849 2860 2870 2898 2908
VARSYM	CONSTANT	77	1742 1754 1954 1974 1994 2016 2659 2679 2699 2719 2794 2797 2835 2848 5030
VARTYP	VARIABLE	2731	2742 2768 2770 2803 2805
WHATTYP	VARIABLE	1343	1393 1393 1394 1394 1396
WHICHTYP	VARIABLE	1110	1112
WHILESYM	CONSTANT	77	4787 4859 4989 5004
WITHSYM	CONSTANT	77	393 484 4788 4921 4989 5004
WITHVARL	PROCEDUR	4704	4747 4950
WORD	VARIABLE	3414	3447 3448 3449 3450
WORD	VARIABLE	3654	3677 3677 3678 3678 3693 3693 3694 3694
WORKSPAC	VARIABLE	299	738 796 796 842 867 878 905 905 925 926 961 984 984 985 986 1468 1490 1491 1494 1496 1497 1500 1509 1509 1537
WRAP	PROCEDUR	966	1471 1603 1614 1623 3123 3271 3275 3279 3284 3308 3310 3312 3450 3497 3679 3695 3809 3815 3918 3924 3932 3938 3985 4057 4063 4083 4089 4229 4231 4233 4238 4240 4242 4244 4262 4270 4279 4285 4318 4326 4336 4343 4348 4350 4362 4365 4367 4425 4444 4450 4454 4461 4506 4807 4828 4831 4839 4842 4877 4880 4883 4885 4888 5105 5113 5147 5158
WRAPNRS	PROCEDUR	964	972 1474
WRITE	PROCEDUR	STANDARD	379 381 738 5443 5456
WRITELN	PROCEDUR	STANDARD	359 361 385 386 387 440 741 743 1115 1117 1120 1121 1122 1126 1127 1130 1133 1136 1137 1154 1156 1157 1158 1159 1161 1163 1164 1167 1169 1176 1183 1184 5184 5186 5455 5458 5467 5469
WRONGID	VARIABLE	406	408
WRONGTK	VARIABLE	397	401
WSPOS	FIELD	183	738 784 792 793 841 867 877 905 905 925 926 936 936 961 984 984 985 986 1464 1466 1487 1509 1509 1537
WSTMNR	VARIABLE	4771	4871 4878 4881 4889
XREF	FIELD	112	1061 1061 1061 1102 1102 1148 1265 1265 1314 1314 1317 1318 1332 1333 1961 1962 1962 2004 2005 2005 2147 2147 2148 2333 2333 2334 2424 2424 2425 2667 2667 2668 2709 2709 2710 2741 2741 2741 2901 2902 2902 3068 3068 3069 3082 3082 3083 3347 3348 4742 4945 5040 5195 5196 5196 5203 5204 5204 5335 5341 5346 5351 5355 5360 5366 5374 5383 5391
XREFLIST	VARIABLE	294	625 5399 5437
XREFS	TYPE	151	152 112 1084 1144 1300 3222 4775
XRFF	VARIABLE	267	1115 1117 1120 1121 1122 1126 1127 1130 1133 1136 1137 1145 1154 1156 1157 1158 1159 1161 1163 1164 1167 1169 1176 1183 1184 1184 5443 5445 5445 5447 5448 5450 5451 5451 5452 5452
XRSCNR	VARIABLE	2731	2741 2769 2804
XRSCNR	VARIABLE	3007	3022 3030 3068 3082
ZEROCNT	VARIABLE	336	343 345 345 346 347 349 351
ZEROSUP	CONSTANT	1478	1490 1491 1494 1496 1497 1500 1510